ORIGINAL

MARK D. FOWLER (Bar No. 124235)
mark.fowler@dlapiper.com
DAVID ALBERTI (Bar. No. 220625)
david.alberti@dlapiper.com
CHRISTINE K. CORBETT (Bar No. 209128)
christine.corbett@dlapiper.com
YAKOV M. ZOLOTOREV (Bar No. 224260)
yakov.zolotorev@dlapiper.com
CARRIE L. WILLIAMSON (Bar No. 230873)
carrie.williamson@dlapiper.com

DLA PIPER US LLP
2000 University Avenue
East Palo Alto, CA 94303-2215
Tel: 650.833.2000
Fax: 650.833.2001

Attorneys for Plaintiff,
Sun Microsystems, Inc.

FILED
MAR 26 2008
RICHARD W. WIEKING
CLERK, U.S. DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

SUN MICROSYSTEMS, INC., a Delaware corporation, Plaintiff, v. NETWORK APPLIANCE, INC., a Delaware corporation, Defendant.	CASE NO. CV 08 1641 EMC **COMPLAINT FOR PATENT INFRINGEMENT** **DEMAND FOR JURY TRIAL**

Plaintiff Sun Microsystems, Inc. ("Sun") complains and alleges as follows against Defendant Network Appliance, Inc. ("NetApp"):

THE PARTIES AND THE NATURE OF THIS ACTION

1. Founded in 1982, Sun has a proven track record of 25 years of innovation. Sun's inventive technology, a result of the intelligence and creativity of its employees and Sun's substantial investment in research and development, is reflected by the over 6,000 United States patents now held by Sun. Sun is serious about innovation – matching words with deeds in investing approximately $1.5 billion dollars in research and development during its last fiscal year alone. In contrast, reflecting its relative lack of innovation, on information and belief,

NetApp, a Sun competitor, spent only about a fourth as much on research and development (approximately $390 million during its last fiscal year) and holds only approximately 200 United States patents. Indeed, rather than innovate, NetApp builds on the innovation of others. For example, while, as alleged below, Sun develops breakthrough software and shares it with open source communities, NetApp, on information and belief, uses extensive amounts of open source code developed by others, without contributing any innovation of its own.

2. Sun provides network computing infrastructure solutions that include computer systems, software, storage and services. Sun's core brands include the Java technology platform, the Solaris operating system, StorageTek and the UltraSPARC processor. Sun's network computing infrastructure solutions are used in a wide range of industries, including the technical and scientific, business, engineering, telecommunications, financial services, manufacturing, retail, government, life sciences, media and entertainment, transportation, energy and utilities, and health care industries.

3. Sun is informed and believes and, on that basis alleges, that on or about January 28, 2008, NetApp acquired Onaro, Inc. ("Onaro"), including Onaro's storage service management software.

4. As alleged below, NetApp's storage service management software, acquired through Onaro, including but not limited to the Onaro SANscreen product suite and NAS Insight software products and networks, storage devices and systems on which those software products operate, uses Sun's innovative patented technology. Because NetApp uses Sun's patented technology without a license to do so, NetApp is infringing Sun's patents, and Sun is entitled to compensation from NetApp for its unauthorized use of Sun's technology, and to an injunction against NetApp to halt its continued use of Sun's technology.

5. Sun is a Delaware corporation having its principal place of business at 4150 Network Circle, Santa Clara, California 95054.

6. Sun is informed and believes and thereon alleges that defendant NetApp is a Delaware corporation having its principal place of business at 495 E. Java Drive, Sunnyvale,

California 94089.

JURISDICTION

7. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331, 1338 and 1367, in that this is a civil action involving claims arising under the law of the United States and claims arising under the Patent and Trademark Act, 35 U.S.C. § 1 *et seq.*

8. This Court has personal jurisdiction over NetApp because it resides in, and transacts business within, the State of California and the Northern District.

VENUE

9. Venue is proper in this district pursuant to 28 U.S.C. § 1400 because NetApp resides in this district and because NetApp has committed acts of infringement in this district and has a regular established place of business in this district. Venue also is proper in this district pursuant to 28 U.S.C. § 1391 because the events or omissions which give rise to this action occurred in this district and NetApp is a corporation that is subject to personal jurisdiction in California and is deemed to reside in this district under § 1391(c).

INTRADISTRICT ASSIGNMENT

10. Sun and NetApp are currently involved in two lawsuits pending in the Northern District of California, San Francisco division, before the Honorable Elizabeth D. Laporte and, therefore, assignment to the San Francisco Division is proper and appropriate. Further, pursuant to Local Rule 3-2(b) and (c), this action may be filed in and assigned to the San Francisco Division as this action includes intellectual property claims and, therefore, may be assigned on a district-wide basis.

THE PATENTS

11. On November 15, 2005, United States Patent No. 6,965,951 (the "'951 patent"), entitled "Device centric discovery and configuration for fabric devices," was duly and legally issued by the United States Patent and Trademark Office (the "USPTO") to Hyon T. Kim. Sun is the owner of the entire right, title and interest in and to the '951 patent. A true and correct copy of the '951 patent is attached hereto as Exhibit A.

12. On November 21, 2000, United States Patent No. 6,151,683 (the "'683 patent"), entitled "Rebuilding computer states remotely," was duly and legally issued by the USPTO to Michael J. Wookey. Sun is the owner of the entire right, title and interest in and to the '683 patent. A true and correct copy of the '683 patent is attached hereto as Exhibit B.

13. On January 30, 2001, United States Patent No. 6,182,249 (the "'249 patent"), entitled "Remote alert monitoring and trend analysis," was duly and legally issued by the USPTO to Michael J. Wookey and Kevin L. Chu. Sun is the owner of the entire right, title and interest in and to the '249 patent. A true and correct copy of the '249 patent is attached hereto as Exhibit C.

14. On November 19, 2002, United States Patent No. 6,484,200 (the "'200 patent"), entitled "Distinguished name scoping system for event filtering," was duly and legally issued by the USPTO to Rajeev Angal, Shivaram Bhat, Michael Roytman and Subodh Bapat. Sun is the owner of the entire right, title and interest in and to the '200 patent. A true and correct copy of the '200 patent is attached hereto as Exhibit D.

FIRST CLAIM FOR RELIEF
(Infringement of the '951 Patent)

15. Sun incorporates and realleges paragraphs 1 through 14 of these claims for relief.

16. Sun is informed and believes, and on that basis alleges, that NetApp is infringing the '951 patent by making, using, offering for sale, and/or selling within the United States devices that embody the inventions disclosed and claimed in the '951 patent, and/or by importing into the United States devices that embody the inventions disclosed and claimed in said patent. On information and belief, NetApp has been and is currently infringing one or more claims of the '951 patent, directly or indirectly, pursuant to 35 U.S.C. §271, in connection with certain of its products, services, methods and/or systems, including without limitation the Onaro SANscreen product suite and NAS Insight software products and networks, storage devices and systems on which those software products operate.

17. In addition to direct infringement, Sun is informed and believes, and on that basis alleges, that NetApp has induced and contributed to infringement by others of the '951 patent.

18. Based upon information and belief, NetApp has notice that it is infringing the '951 patent. Despite such notice, NetApp has continued to willfully infringe said patent by making, using, offering to sell, and/or selling within the United States products that embody the inventions disclosed and claimed in said patent, and/or by importing such products into the United States.

19. Sun has been irreparably harmed by NetApp's acts of infringement, and will continue to be harmed unless and until NetApp's acts of infringement are enjoined and restrained by order of this Court. Sun has no adequate remedy at law and is entitled to a preliminary and permanent injunction against NetApp and its infringing products.

20. As a result of NetApp's acts of infringement, Sun has suffered and will continue to suffer damages in an amount to be proven at trial.

21. This case is an "exceptional" case within the meaning of 35 U.S.C. § 285 and Sun is entitled to an award of attorneys' fees.

SECOND CLAIM FOR RELIEF
(Infringement of the '683 Patent)

22. Sun incorporates and realleges paragraphs 1 through 14 of these claims for relief.

23. Sun is informed and believes, and on that basis alleges, that NetApp is infringing the '683 patent by making, using, offering for sale, and/or selling within the United States devices that embody the inventions disclosed and claimed in the '683 patent, and/or by importing into the United States devices that embody the inventions disclosed and claimed in said patent. On information and belief, NetApp has been and is currently infringing one or more claims of the '683 patent, directly or indirectly, pursuant to 35 U.S.C. §271, in connection with certain of its products, services, methods and/or systems, including without limitation including without limitation the Onaro SANscreen product suite and NAS Insight software products and networks, storage devices and systems on which those software products operate.

24. In addition to direct infringement, Sun is informed and believes, and on that basis alleges, that NetApp has induced and contributed to infringement by others of the '683 patent.

25. Based upon information and belief, NetApp has notice that it is infringing the '683

patent. Despite such notice, NetApp has continued to willfully infringe said patent by making, using, offering to sell, and/or selling within the United States products that embody the inventions disclosed and claimed in said patent, and/or by importing such products into the United States.

26. Sun has been irreparably harmed by NetApp's acts of infringement, and will continue to be harmed unless and until NetApp's acts of infringement are enjoined and restrained by order of this Court. Sun has no adequate remedy at law and is entitled to a preliminary and permanent injunction against NetApp and its infringing products.

27. As a result of NetApp's acts of infringement, Sun has suffered and will continue to suffer damages in an amount to be proven at trial.

28. This case is an "exceptional" case within the meaning of 35 U.S.C. § 285 and Sun is entitled to an award of attorneys' fees.

THIRD CLAIM FOR RELIEF
(Infringement of the '249 Patent)

29. Sun incorporates and realleges paragraphs 1 through 14 of these claims for relief.

30. Sun is informed and believes, and on that basis alleges, that NetApp is infringing the '249 patent by making, using, offering for sale, and/or selling within the United States devices that embody the inventions disclosed and claimed in the '249 patent, and/or by importing into the United States devices that embody the inventions disclosed and claimed in said patent. On information and belief, NetApp has been and is currently infringing one or more claims of the '249 patent, directly or indirectly, pursuant to 35 U.S.C. §271, in connection with certain of its products, services, methods and/or systems, including without limitation the Onaro SANscreen product suite and NAS Insight software products and networks, storage devices and systems on which those software products operate.

31. In addition to direct infringement, Sun is informed and believes, and on that basis alleges, that NetApp has induced and contributed to infringement by others of the '249 patent.

32. Based upon information and belief, NetApp has notice that it is infringing the '249 patent. Despite such notice, NetApp has continued to willfully infringe said patent by making,

using, offering to sell, and/or selling within the United States products that embody the inventions disclosed and claimed in said patent, and/or by importing such products into the United States.

33. Sun has been irreparably harmed by NetApp's acts of infringement, and will continue to be harmed unless and until NetApp's acts of infringement are enjoined and restrained by order of this Court. Sun has no adequate remedy at law and is entitled to a preliminary and permanent injunction against NetApp and its infringing products.

34. As a result of NetApp's acts of infringement, Sun has suffered and will continue to suffer damages in an amount to be proven at trial.

35. This case is an "exceptional" case within the meaning of 35 U.S.C. § 285 and Sun is entitled to an award of attorneys' fees.

FOURTH CLAIM FOR RELIEF
(Infringement of the '200 Patent)

36. Sun incorporates and realleges paragraphs 1 through 14 of these claims for relief.

37. Sun is informed and believes, and on that basis alleges, that NetApp is infringing the '200 patent by making, using, offering for sale, and/or selling within the United States devices that embody the inventions disclosed and claimed in the '200 patent, and/or by importing into the United States devices that embody the inventions disclosed and claimed in said patent. On information and belief, NetApp has been and is currently infringing one or more claims of the '200 patent, directly or indirectly, pursuant to 35 U.S.C. §271, in connection with certain of its products, services, methods and/or systems, including without limitation the Onaro SANscreen product suite and NAS Insight software products and networks, storage devices and systems on which those software products operate.

38. In addition to direct infringement, Sun is informed and believes, and on that basis alleges, that NetApp has induced and contributed to infringement by others of the '200 patent.

39. Based upon information and belief, NetApp has notice that it is infringing the '200 patent. Despite such notice, NetApp has continued to willfully infringe said patent by making, using, offering to sell, and/or selling within the United States products that embody the inventions

disclosed and claimed in said patent, and/or by importing such products into the United States.

40. Sun has been irreparably harmed by NetApp's acts of infringement, and will continue to be harmed unless and until NetApp's acts of infringement are enjoined and restrained by order of this Court. Sun has no adequate remedy at law and is entitled to a preliminary and permanent injunction against NetApp and its infringing products.

41. As a result of NetApp's acts of infringement, Sun has suffered and will continue to suffer damages in an amount to be proven at trial.

42. This case is an "exceptional" case within the meaning of 35 U.S.C. § 285 and Sun is entitled to an award of attorneys' fees.

PRAYER

WHEREFORE: Sun prays for judgment against NetApp as follows:

1. A judgment that NetApp has infringed, induced others to infringe, and/or committed acts of contributory infringement with respect to the claims of each of the Sun Patents-in-Suit;

2. A judgment that NetApp's patent infringement has been, and continues to be, willful and deliberate;

3. An order preliminarily and permanently enjoining NetApp and its subsidiaries, officers, agents, servants, employees, licensees and all other persons acting or attempting to act in active concert or participation with it or acting on its behalf, from further infringement, inducement or infringement, or contributory infringement of each of the Sun Patents-in-Suit;

4. An order directing NetApp to account for and pay to Sun all damages caused to Sun by reason of NetApp's patent infringement, pursuant to 35 U.S.C. § 284, including increased damages under 35 U.S.C. § 284;

5. An order directing NetApp to pay Sun's costs, expenses and reasonable attorneys' fees pursuant to 35 U.S.C. § 285;

6. An award of pre-judgment and post-judgment interest on the damages caused to Sun by NetApp;

7. For the costs incurred by Sun in bringing this suit; and

8. For other and further legal and/or equitable relief as the Court deems just and proper.

Dated: March 26, 2008

DLA PIPER US LLP

Christine K. Corbett

Mark D. Fowler
David Alberti
Christine K. Corbett
Yakov M. Zolotorev
Carrie L. Williamson
Attorneys for Plaintiff
SUN MICROSYSTEMS, INC.

DEMAND FOR JURY TRIAL

Sun demands trial by jury for all so triable pursuant to Fed. R. Civ. Pro. 38(b) and Civil L.R. 3-6(a).

Dated: March 26, 2008

DLA PIPER US LLP

Christine K. Corbett

Mark D. Fowler
David Alberti
Christine K. Corbett
Yakov M. Zolotorev
Carrie L. Williamson
Attorneys for Plaintiff
SUN MICROSYSTEMS, INC.

EXHIBIT A

US006965951B2

(12) **United States Patent**
Kim

(10) **Patent No.: US 6,965,951 B2**
(45) **Date of Patent: Nov. 15, 2005**

(54) **DEVICE CENTRIC DISCOVERY AND CONFIGURATION FOR FABRIC DEVICES**

(75) Inventor: **Hyon T. Kim**, San Jose, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 374 days.

(21) Appl. No.: **10/147,591**

(22) Filed: **May 17, 2002**

(65) **Prior Publication Data**

US 2003/0217212 A1 Nov. 20, 2003

(51) **Int. Cl.**[7] **G06F 3/00**; G06F 15/177
(52) **U.S. Cl.** **710/19**; 709/220
(58) **Field of Search** 710/17–19, 104; 709/220, 221, 222, 223, 224

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,600,791 A		2/1997	Carlson et al.	
5,805,924 A		9/1998	Stoevhase	
5,872,932 A		2/1999	Schettler et al.	
5,941,972 A		8/1999	Hoese et al.	
5,944,798 A		8/1999	McCarty et al.	
5,959,994 A		9/1999	Boggs et al.	
5,974,546 A		10/1999	Anderson	
6,009,466 A	*	12/1999	Axberg et al.	709/220
6,016,144 A		1/2000	Blonstein et al.	
6,182,167 B1		1/2001	Basham et al.	
6,199,112 B1		3/2001	Wilson	
6,229,540 B1		5/2001	Tonelli et al.	
6,304,549 B1		10/2001	Srinivasan et al.	
6,344,862 B1		2/2002	Williams et al.	
6,393,489 B1		5/2002	Sambamurthy et al.	
6,446,141 B1		9/2002	Nolan et al.	
6,449,734 B1		9/2002	Shrivastava et al.	
6,473,405 B2		10/2002	Ricciulli	
6,538,669 B1		3/2003	Lagueux, Jr. et al.	
6,584,499 B1		6/2003	Jantz et al.	
6,594,698 B1		7/2003	Chow et al.	
6,633,538 B1		10/2003	Tanaka et al.	
6,640,278 B1		10/2003	Nolan	
6,643,748 B1		11/2003	Wieland	
6,654,752 B2		11/2003	Ofek	
6,665,714 B1		12/2003	Blumenau et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

EP	989 490	3/2000
EP	1 085 414	3/2001
WO	98/18306	5/1998
WO	01/14987	3/2001

OTHER PUBLICATIONS

Khattar, et al., "Introduction to Storage Area Network, SAN," IBM, SG24-5470-00, International Technical Support Organization, Aug. 1999, 54 pages.

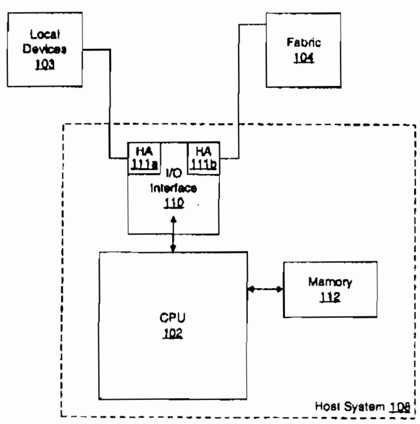
(Continued)

Primary Examiner—Khanh Dang
(74) *Attorney, Agent, or Firm*—Robert C. Kowert; Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.

(57) **ABSTRACT**

A host may be coupled to a fabric network. Fabric devices attached to the fabric network may be visible to the host through one or more host adapter ports. The host system may include a device centric discovery interface configured to provide an interface to a fabric driver to obtain information about the devices in the fabric network. The device centric discovery interface may be configured to return device centric discovery information such that a multi-path fabric device is presented as a single device with transport information provided for each path to the multi-path device. A device centric configuration interface may provide an interface to the fabric driver for device centric configuration of the devices in the fabric for use by the host such that a requested fabric device is configured for use by the host on multiple paths in the fabric network.

22 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

6,683,605	B1	1/2004	Bi et al.
6,694,361	B1	2/2004	Shah et al.
6,728,789	B2	4/2004	Odenwald et al.
6,748,459	B1	6/2004	Lin et al.
6,792,479	B2	9/2004	Allen et al.
2001/0047482	A1	11/2001	Harris et al.

OTHER PUBLICATIONS

"Open SANs, An In-Dept Brief,"Version 1.1, Dec. 2000, 10 pages.

American National Standards Institute *for Information Technology* "Fibre Channel—General Services—3 (FC-GS-3)," ANSI NCITS 348-2001, 2001 Information Technology Industry Council, 261 pages.

American National Standards Institute *for Information Technology* "Fibre Channel—Switch Fabric (FC-SW)," ANSI NCITS 321-1998, 1998 Information Technology Industry Council, 108 pages.

American National Standards Institute *for Information Technology* "Fibre Channel Arbitrated Loop (FC-AL-2)," ANSI NCITS 332-1999, 1999 Information Technology Industry Council, 149 pages.

American National Standards Institute *for Information Technology* "Fibre Channel—Arbitrated Loop (FC-AL)," ANSI X3.272-1996, 1996 Information Technology Industry Council, 102 pages.

American National Standards Institute *for Information Technology* "Fibre Channel—Fabric Generic Requirements (FC-FG)," ANSI X3.289-1996, 1997 Information Technology Industry Council, 33 pages.

* cited by examiner

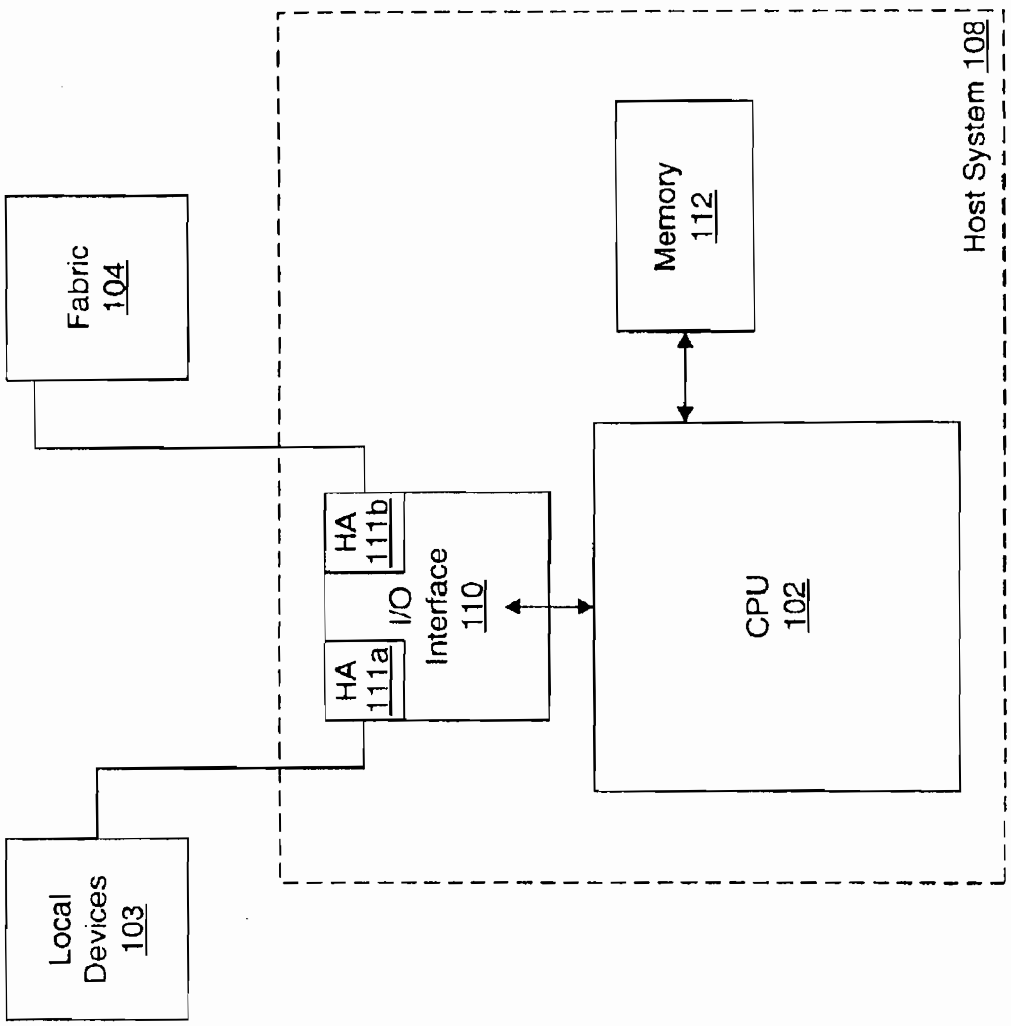

Fabric 104
Local Devices 103
HA 111a
HA 111b
I/O Interface 110
CPU 102
Memory 112
Host System 108

FIGURE 1

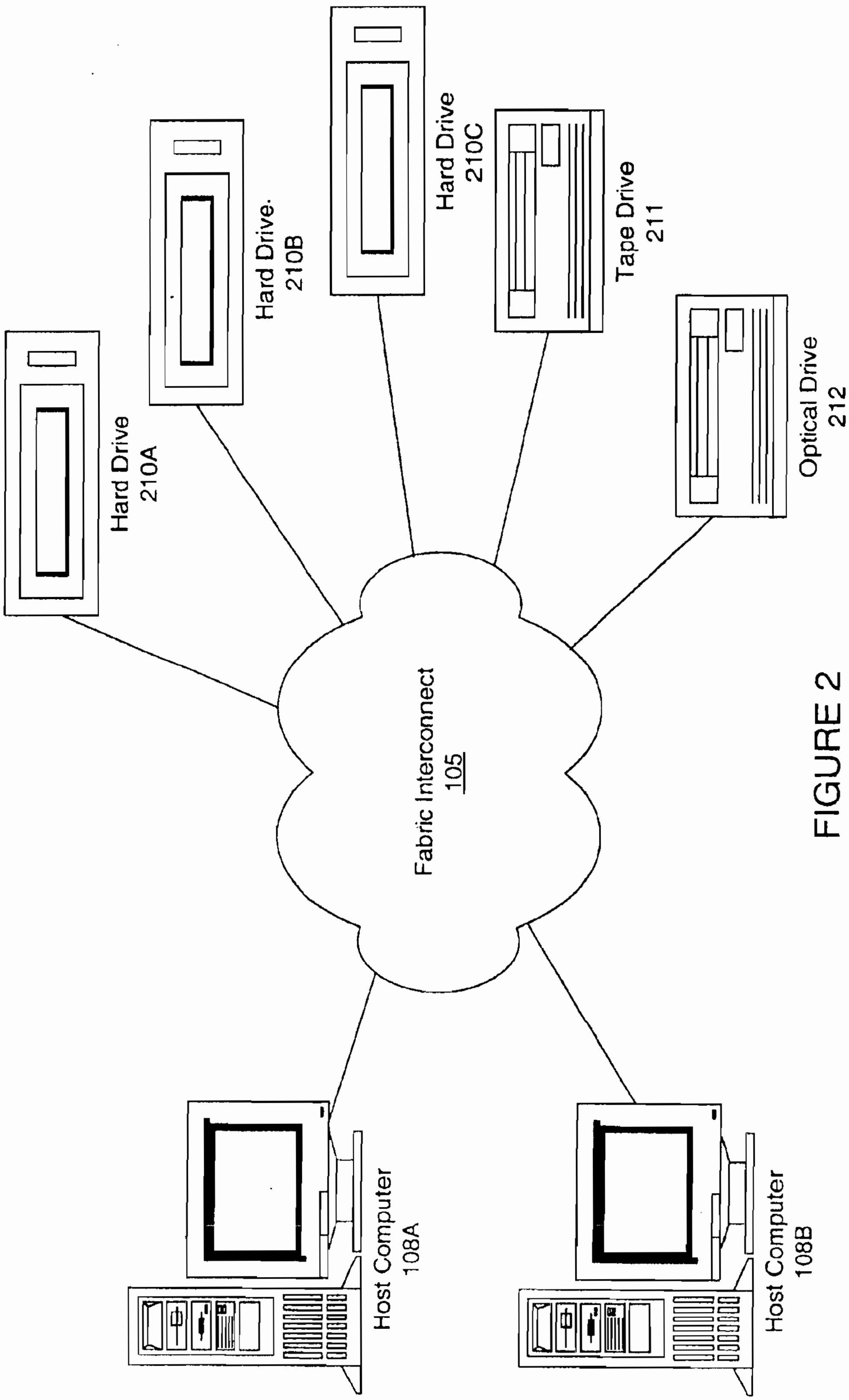

Hard Drive
210A
Hard Drive
210B
Hard Drive
210C
Tape Drive
211
Optical Drive
212
Fabric Interconnect
105
Host Computer
108A
Host Computer
108B

FIGURE 2

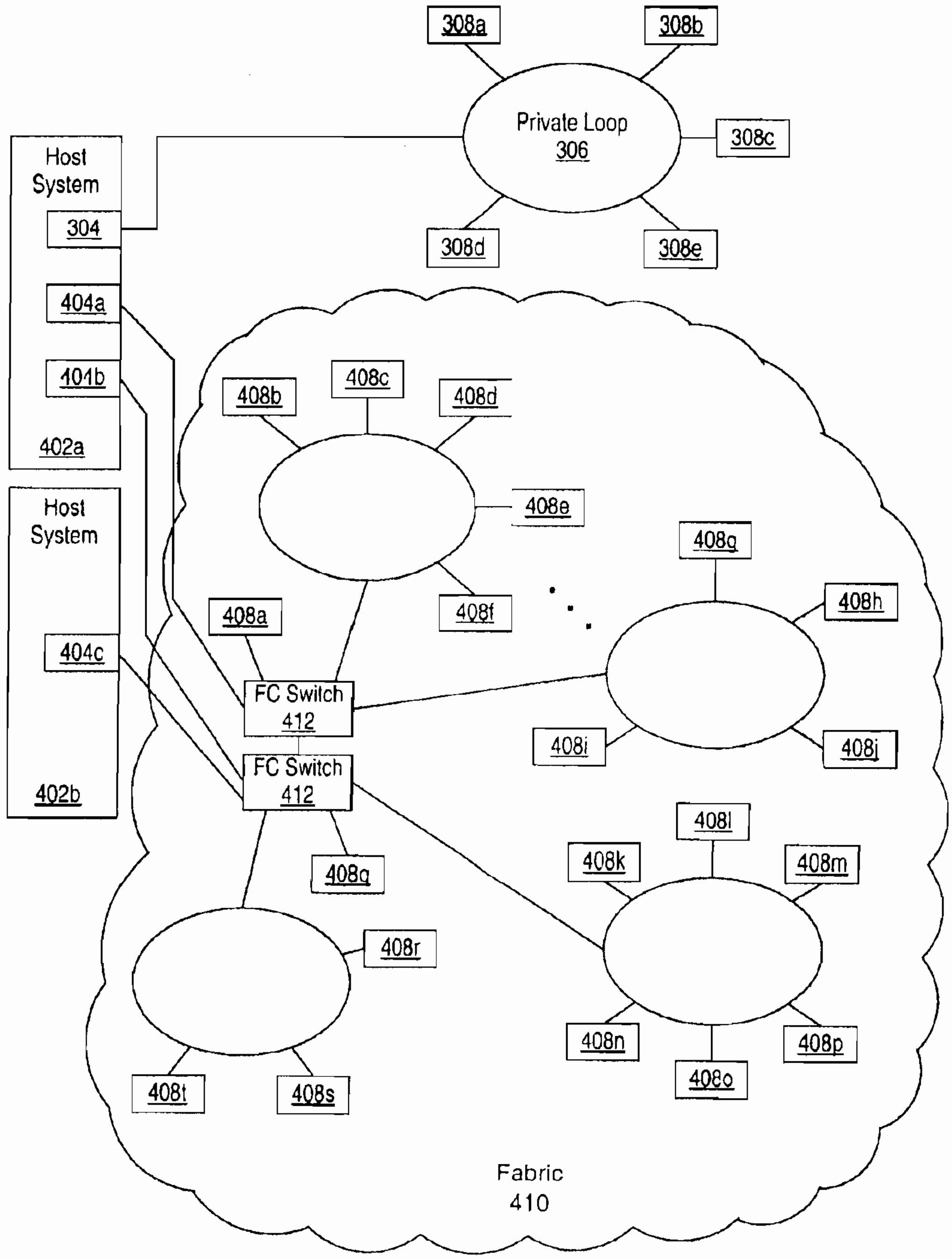

308a
308b
Private Loop
306
308c
308d
308e
Host
System
304
404a
404b
402a
Host
System
404c
402b
408b
408c
408d
408e
408g
408h
408a
408f
FC Switch
412
FC Switch
412
408i
408j
408l
408k
408m
408q
408r
408n
408o
408p
408t
408s
Fabric
410

FIGURE 3

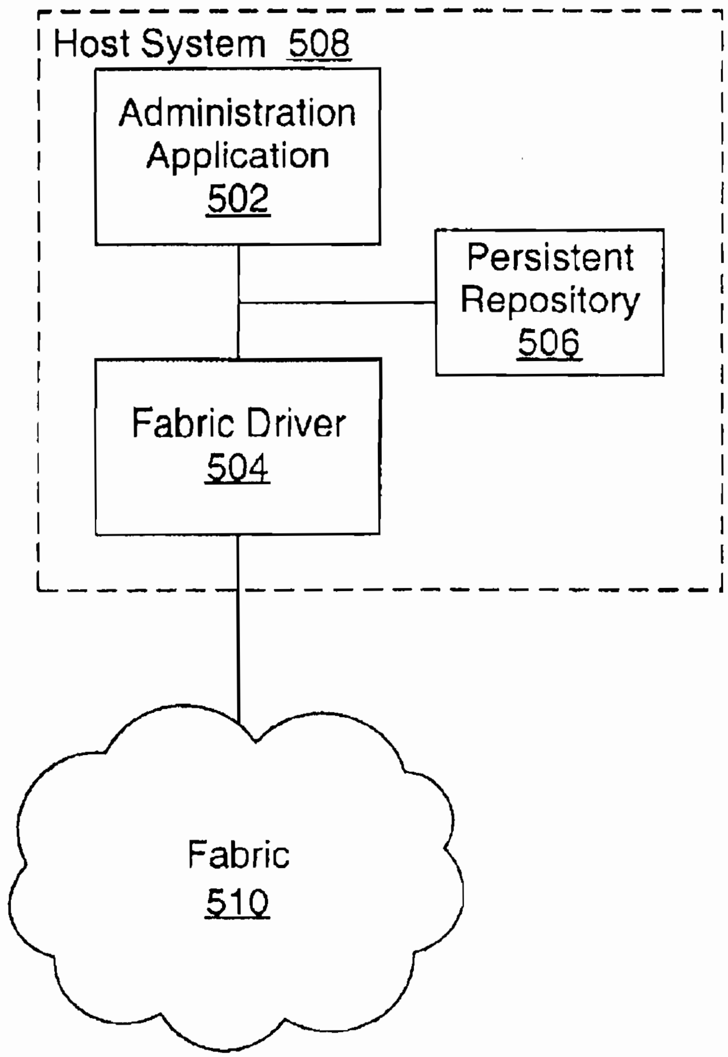

Host System 508
Administration Application 502
Persistent Repository 506
Fabric Driver 504
Fabric 510

FIGURE 4

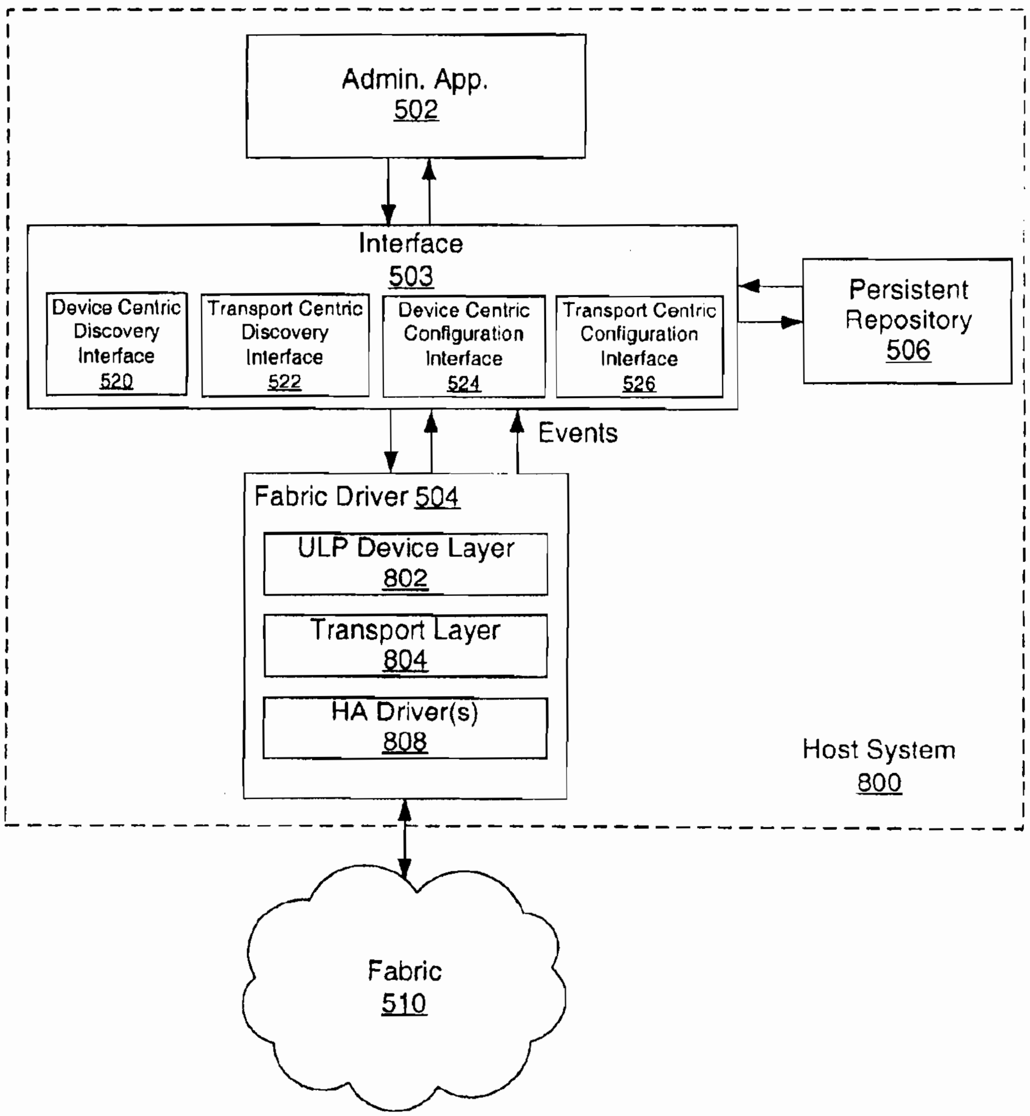

Admin. App.
502
Interface
503
Device Centric Discovery Interface
520
Transport Centric Discovery Interface
522
Device Centric Configuration Interface
524
Transport Centric Configuration Interface
526
Persistent Repository
506
Events
Fabric Driver 504
ULP Device Layer
802
Transport Layer
804
HA Driver(s)
808
Host System
800
Fabric
510

FIGURE 5

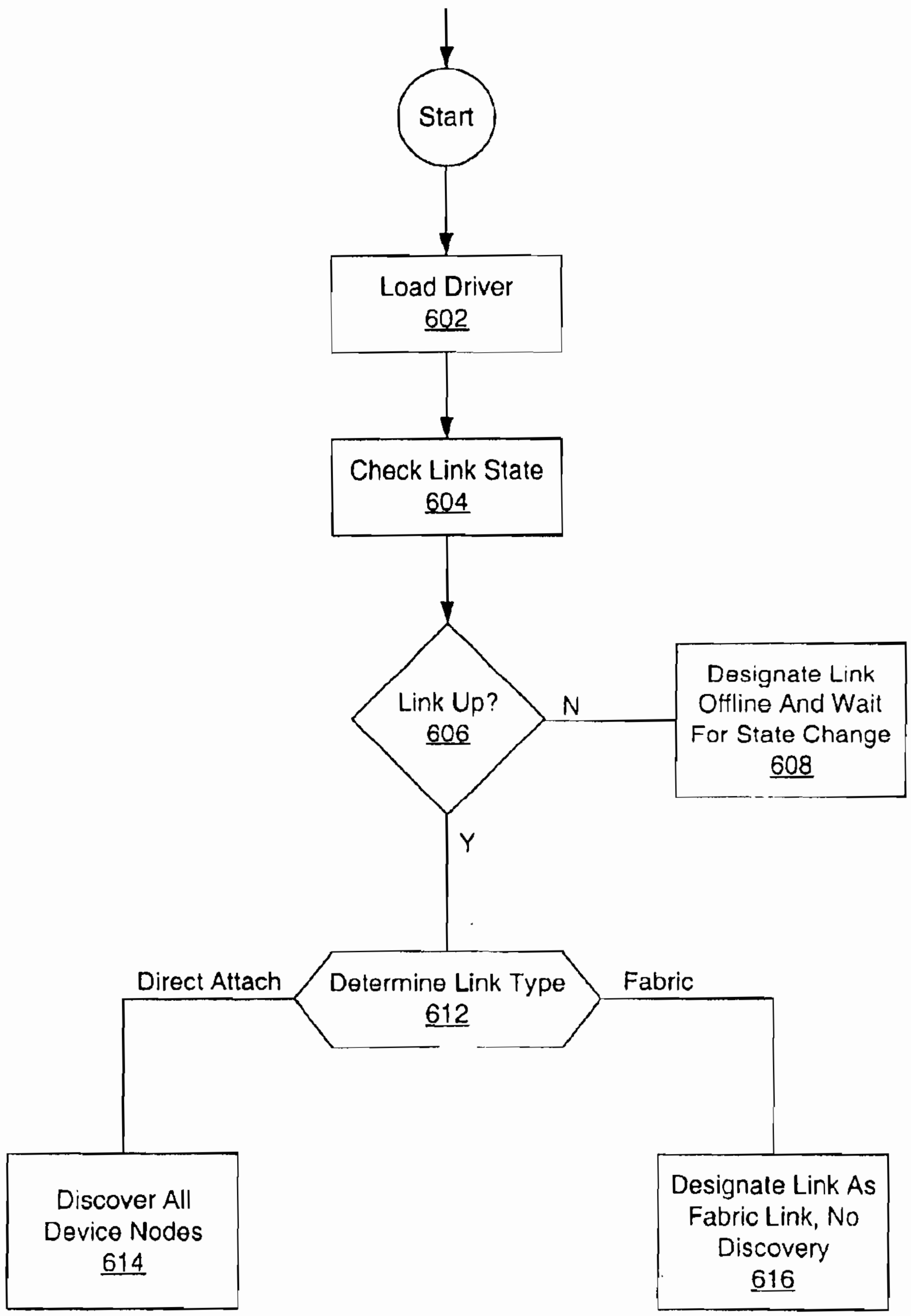

Start
Load Driver
602
Check Link State
604
Link Up?
606
N
Designate Link Offline And Wait For State Change
608
Y
Direct Attach
Determine Link Type
612
Fabric
Discover All Device Nodes
614
Designate Link As Fabric Link, No Discovery
616

FIGURE 6

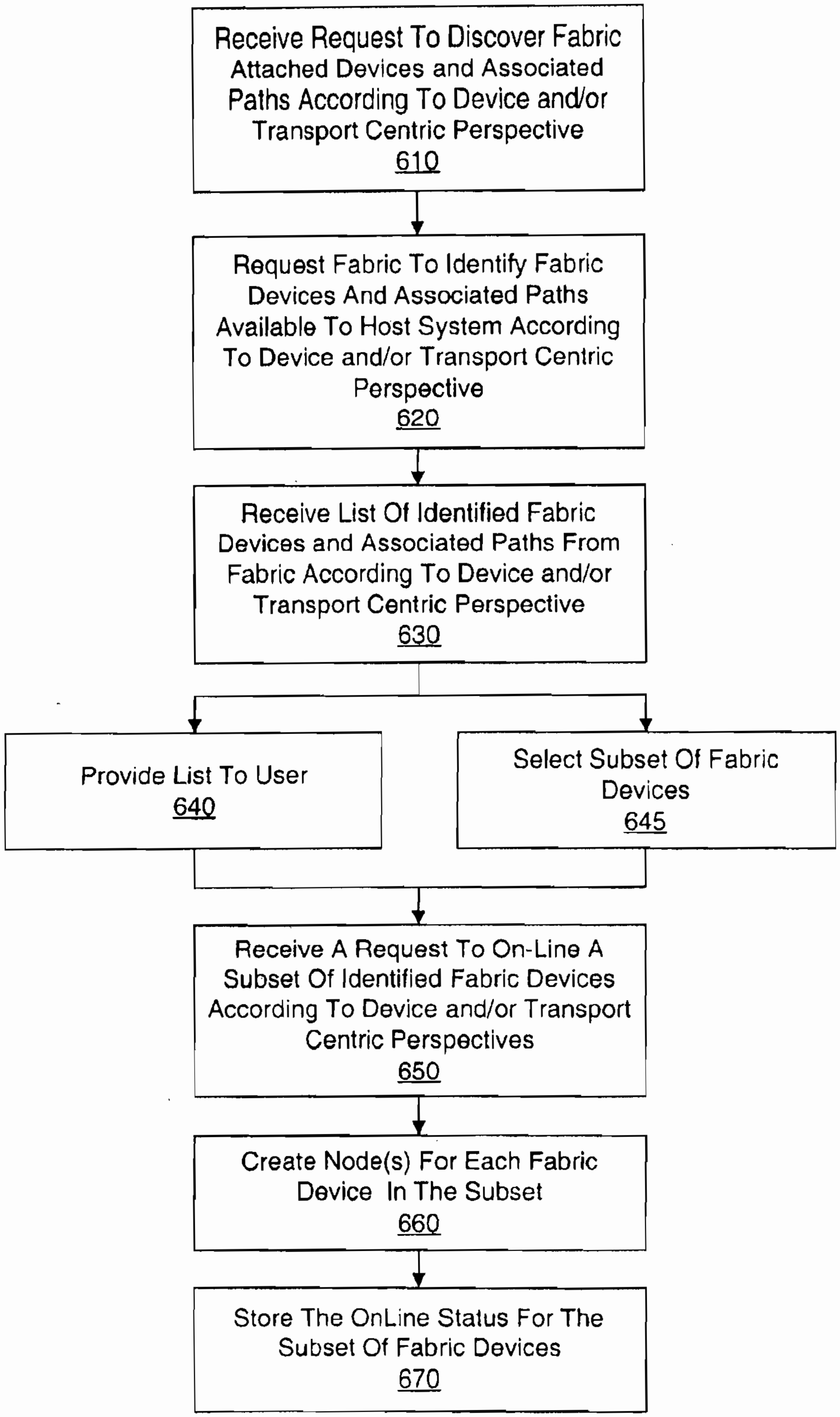

Receive Request To Discover Fabric Attached Devices and Associated Paths According To Device and/or Transport Centric Perspective
610
Request Fabric To Identify Fabric Devices And Associated Paths Available To Host System According To Device and/or Transport Centric Perspective
620
Receive List Of Identified Fabric Devices and Associated Paths From Fabric According To Device and/or Transport Centric Perspective
630
Provide List To User
640
Select Subset Of Fabric Devices
645
Receive A Request To On-Line A Subset Of Identified Fabric Devices According To Device and/or Transport Centric Perspectives
650
Create Node(s) For Each Fabric Device In The Subset
660
Store The OnLine Status For The Subset Of Fabric Devices
670

FIGURE 7

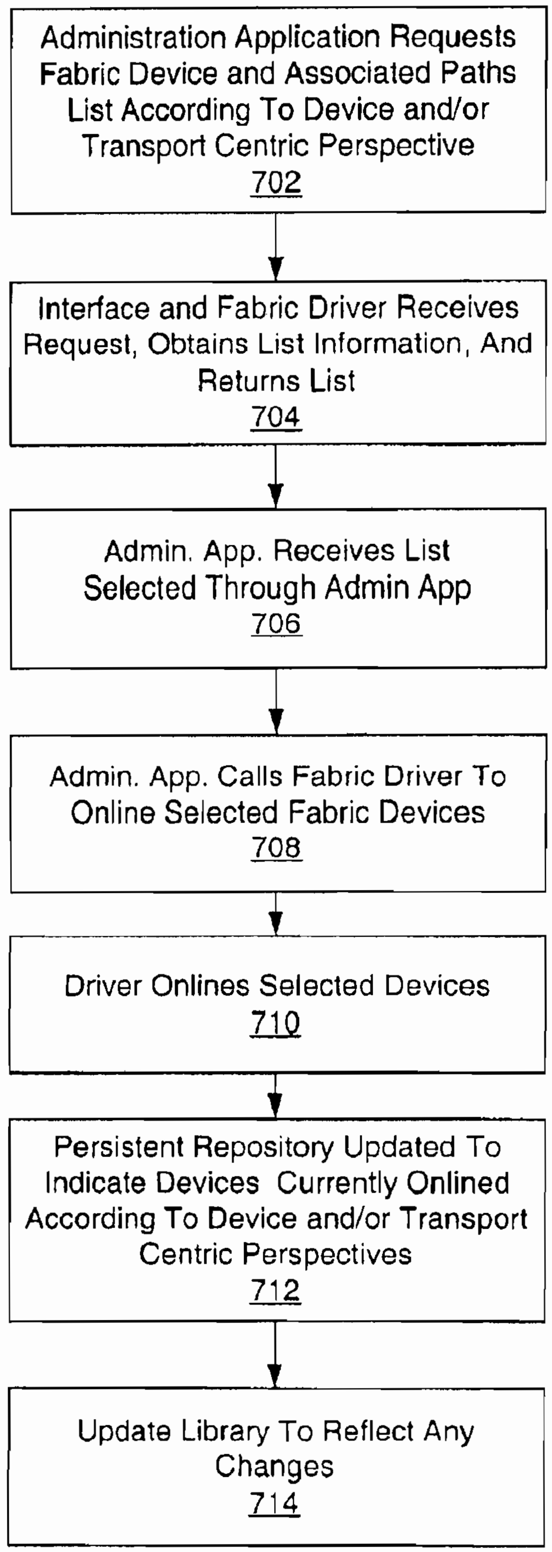

Administration Application Requests Fabric Device and Associated Paths List According To Device and/or Transport Centric Perspective
702
Interface and Fabric Driver Receives Request, Obtains List Information, And Returns List
704
Admin. App. Receives List Selected Through Admin App
706
Admin. App. Calls Fabric Driver To Online Selected Fabric Devices
708
Driver Onlines Selected Devices
710
Persistent Repository Updated To Indicate Devices Currently Onlined According To Device and/or Transport Centric Perspectives
712
Update Library To Reflect Any Changes
714

FIGURE 8

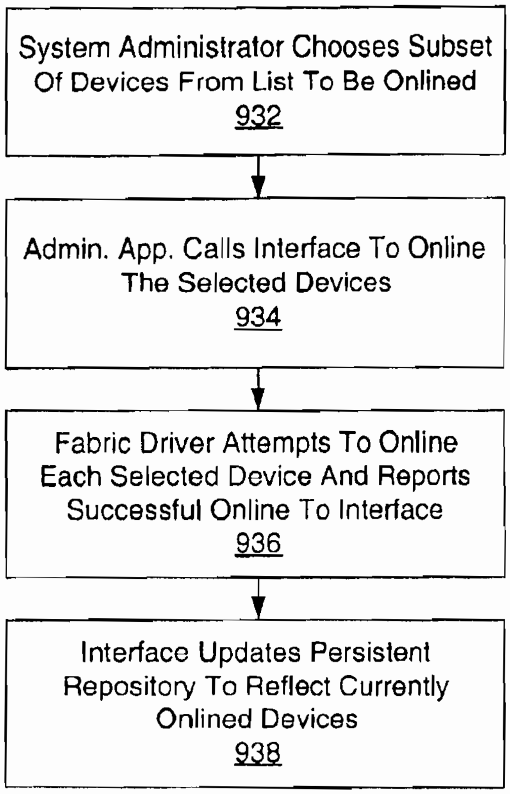


FIGURE 9

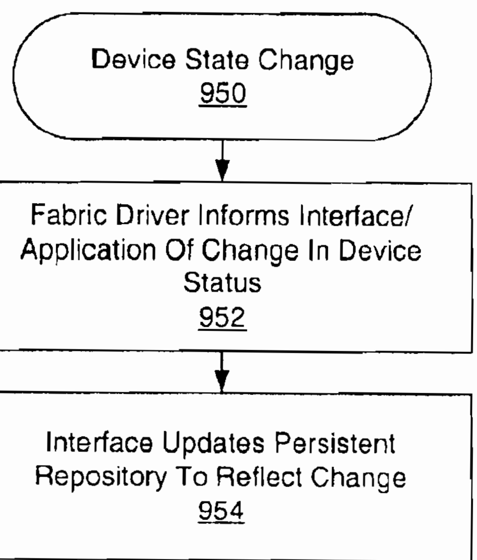


FIGURE 10

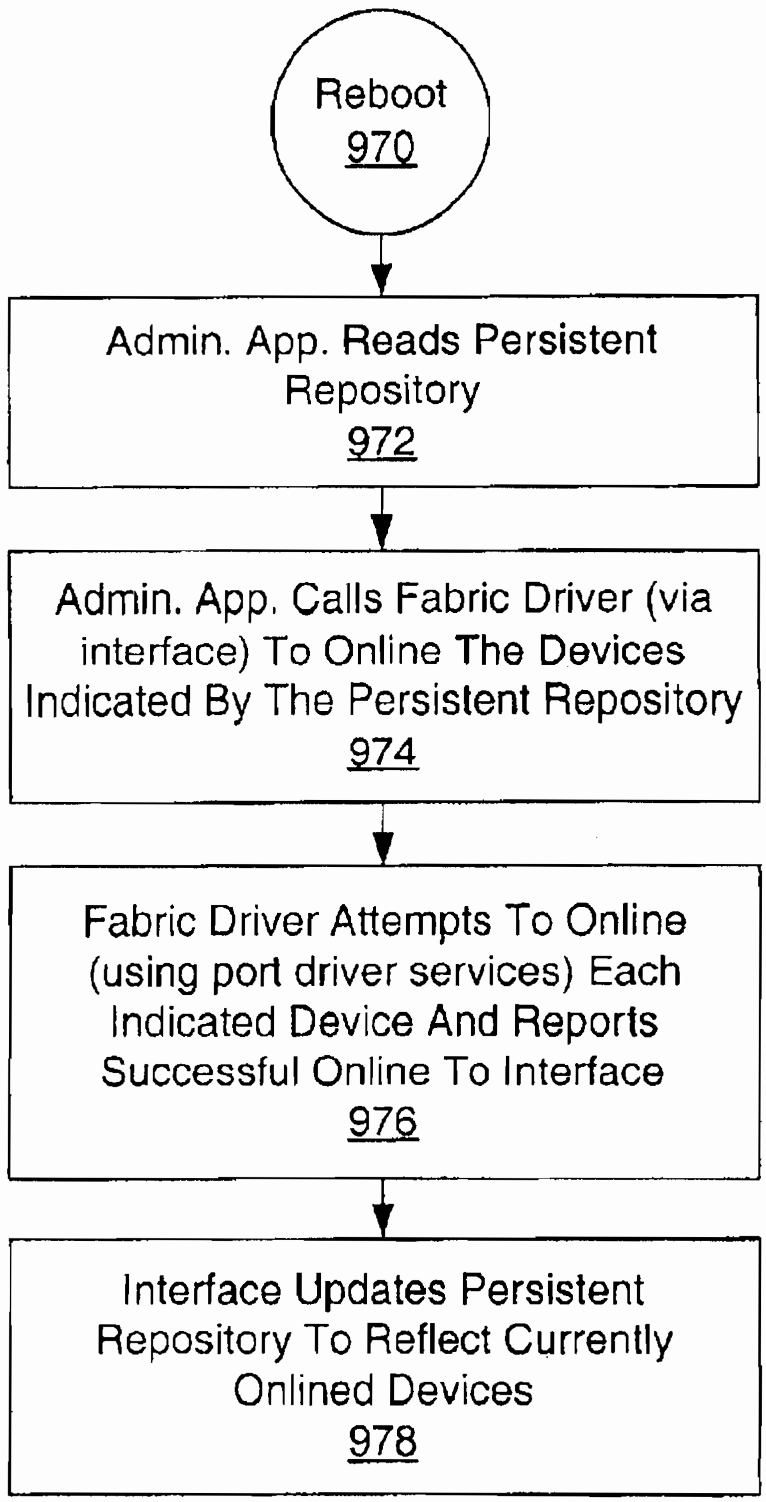

Reboot
970
Admin. App. Reads Persistent Repository
972
Admin. App. Calls Fabric Driver (via interface) To Online The Devices Indicated By The Persistent Repository
974
Fabric Driver Attempts To Online (using port driver services) Each Indicated Device And Reports Successful Online To Interface
976
Interface Updates Persistent Repository To Reflect Currently Onlined Devices
978

FIGURE 11

DEVICE CENTRIC DISCOVERY AND CONFIGURATION FOR FABRIC DEVICES

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to network systems, and more particularly to discovery and configuration of devices attached to a fabric in a storage network.

2. Description of the Related Art

Storage area networks, also referred to as SANs, are dedicated networks that connect one or more systems to storage devices and subsystems. Today, fibre channel is one of the leading technologies for SANs. In general, fibre channel encompasses three networking topologies: point-to-point, loop, and fabric. In a point-to-point topology, a fibre channel host adapter in a system is typically connected to a single fibre channel storage subsystem. In a fibre channel loop network, also called an arbitrated loop, the loop is constructed by connecting devices together in a single logical ring. Loops can be constructed by connecting devices through a fibre channel hub in a star-wired topology or by connecting them together in a connected physical loop from device to device. In a fibre channel fabric topology, the storage networks are constructed with network switches. A fabric can be composed of a single switch or multiple switches. Ports on fabric networks connect devices to switches on low-latency, point-to-point connections.

The devices connected in the loop and fabric topologies may be referred to as "network nodes" and may be any entity that is able to send or receive transmissions in a fibre channel network. For example, a network node may be a computer system, a storage device/subsystem, a storage router/bridge that connects SCSI equipment, a printer, a scanner, or any other equipment, such as data capture equipment. The ANSI X3.272-1996 specification entitled "FC-AL, Fibre Channel Arbitrated Loop" and the ANSI X3.T11 Project 1133-D specification entitled "FC-AL-2, Fibre Channel Arbitrated Loop" describe examples of fibre channel loop topologies in further detail. The ANSI X3.T11 Project 959-D specification entitled "FC-SW Fibre Channel Switch Fabric" describes an example of a fibre channel fabric in further detail. Note that the most recent versions of these and related specifications may be obtained from the T11 technical committee of the National Committee for Information Technology Standards (NCITS).

For point-to-point topologies and loop topologies, device drivers executing on a host computer may perform device discovery at host boot-up to determine locally connected devices. The discovered devices are configured to be accessible to applications running on the host by creating a node for each device within the host. These types of nodes are referred to as operating system device nodes. Each node functions as an internal (to the host) representation of an attached device and provides a communication path to the device. For fabric topologies, discovering fabric devices and associated paths available to the host computer as part of the boot-up process may not be feasible because of the number of devices capable of being attached to the fabric. In addition, there may be multiple paths from a host computer to a particular device in the fabric.

SUMMARY OF THE INVENTION

A host system may have one or more host adapter ports for coupling the system to a fabric network. Fabric devices attached to the fabric network may be visible to the system through the one or more host adapter port. The host system may include a fabric driver configured to interface the system to the fabric network through the host adapter ports. A device centric discovery interface may provide an interface to the fabric driver to obtain information about the devices in the fabric network. The device centric discovery interface may return device centric discovery information such that a multi-path fabric device is presented as a single device with transport information provided for each path to the multi-path device. A single device may be an upper level protocol ULP device, such as a SCSI LUN device wherein a number of ULP devices may share the same fabric transport path. The system may also include a device centric configuration interface for providing an interface to the fabric driver for device centric configuration of the devices (including multi-path devices) in the fabric for use by the host system such that a requested fabric device is configured for use by the host system on multiple paths in the fabric network.

The host system may also include a transport centric discovery interface configured to provide an interface to the fabric driver to obtain information about the devices in the fabric network. The transport centric discovery interface may return transport centric discovery information such that a multi-path device is presented as a separate device for each path to the multi-path device. The host system may also include a transport centric configuration interface for providing an interface to the fabric driver for transport centric configuration of the devices in the fabric for use by the system such that fabric device are configured for use by the system on a requested fabric transport path.

A method may include discovering a plurality of devices on a fabric network and displaying information describing the plurality of discovered devices in a device-centric format, wherein in the device-centric format each device is displayed as a single device with one or more associated paths each corresponding to a particular transport connection to a host. A displayed device may be an upper level protocol device, such as a SCSI LUN device. The method may include configuring a selected one of the devices for use by the host on one or more associated paths in response to user input.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a host computer attached to a fabric and one or more local devices;

FIG. 2 illustrates an example of a storage area network (SAN) suitable for implementing various embodiments;

FIG. 3 is an example of a storage network suitable for implementing various embodiments;

FIG. 4 is an illustration of a host computer coupled to a fabric according to one embodiment;

FIG. 5 is an illustration of a host computer system coupled to a fabric according to one embodiment;

FIG. 6 is an example of a device discovery process according to an embodiment;

FIG. 7 is a flowchart of an on-demand device node creation process, according to one embodiment;

FIG. 8 is a flowchart of an on-demand node creation process for fibre channel fabrics according to an embodiment;

FIG. 9 is a flowchart showing a method for onlining selected fabric devices on-demand from a list of fabric devices according to one embodiment;

FIG. **10** is a flowchart illustrating a mechanism for dynamically updating a persistent repository to reflect changes in the fabric according to one embodiment, and;

FIG. **11** is a flowchart illustrating a mechanism to allow a host's fabric device configuration to persist across reboots and shutdowns according to one embodiment.

While the invention is described herein by way of example for several embodiments and illustrative drawings, those skilled in the art will recognize the invention is not limited to the embodiments or drawings described. It should be understood that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the invention as defined by the appended claims. The headings used herein are for organizational purposes only and are not meant to be used to limit the scope of the description or the claims. As used throughout this application, the word "may" is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words "include", "including", and "includes" mean including, but not limited to.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

Suitable for implementing various embodiments, FIG. **1** illustrates a host system **108** attached to a fabric **104**. The host system may include at least one central processing unit (CPU) or processor **102**. The CPU **102** may be coupled to a memory **112**. The memory **112** is representative of various types of possible memory media, also referred to as "computer readable media". Hard disk storage, floppy disk storage, removable disk storage, flash memory or random access memory (RAM) are examples of memory media. The terms "memory" and "memory medium" may include an installation medium, e.g., a CD-ROM or floppy disk, a computer system memory such as DRAM, SRAM, EDO RAM, SDRAM, DDR SDRAM, Rambus RAM, etc., or a non-volatile memory such as a magnetic media, e.g., a hard drive or optical storage. The memory medium may include other types of memory as well, or combinations thereof. In addition, the memory medium may be located in a first computer in which the programs are executed, or may be located in a second different computer which connects to the first computer over a network. In the latter instance, the second computer may provide the program instructions to the first computer for execution.

The memory **112** may permit two-way access: readable and writable. The memory **112** may store instructions and/or data which implement all or part of the system and method described in detail herein, and the memory **112** may be utilized to install the instructions and/or data. The host system **108** may be any of the various types of devices, including, but not limited to, a personal computer system, desktop computer, laptop computer, palmtop computer, mainframe computer system, workstation, network appliance, network computer, Internet appliance, personal digital assistant (PDA), embedded device, smart phone, television system, or other suitable device. In general, the term "computer system" may be broadly defined to encompass any device having a processor **102** which executes instructions from a memory medium.

The host system **108** may be coupled to a fabric **104**, which may provide access to a plurality of fabric attached devices, such as persistent storage devices or other computer peripheral devices. The CPU **102** may acquire instructions and/or data through an input/output (I/O) interface **110**. Through the input/output interface **110**, the CPU **102** may also be coupled to one or more local devices **103**, such as local input/output devices (video monitors or other displays, track balls, mice, keyboards, etc.) local storage devices (hard drives, optical storage devices, etc.), local printers, plotters, scanners, and any other type of local I/O devices for use with a host system **108**. Some local devices **103** may be referred to as direct attach devices. The input/output interface **110** may include host adapters (HA) **111*a*** and **111*b*** for coupling to the local devices **103** and fabric respectively. Host adapters **111*a*** and **111*b*** may be fibre channel adapters (FCAs). In one embodiment, one or more of the local devices **103** may be included in the host system **108**, such as in expansion slots of the host system **108**. In one embodiment, one or more of the local devices **103** may be externally connected to the host system **108**.

The host system **108** may be able to execute one or more computer programs. The computer programs may comprise an operating system (OS) or other system software, application software, utility software, Java™ applets, and/or any other sequence of instructions. Typically, an operating system performs basic tasks such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers. Application software runs on top of the operating system and provides additional functionality. In one embodiment, the OS may be based on the Solaris™ operating system from Sun Microsystems, Inc. The computer programs may be stored in a memory medium or storage medium such as the memory **112**. Alternatively, the computer programs may be provided to the CPU **102** through the fabric or input/output interface **110**.

FIG. **2** illustrates an example of a storage area network (SAN) coupled to host computers **108**A and **108**B. The SAN includes a fabric interconnect **105** coupled to hard drives **210**A, **210**B, and **210**C, tape drive **211**, and optical drive **212**. Hard drives **210**A, **210**B, and **210**C, tape drive **211**, and optical drive **212** may also be referred to as fabric devices. Each device may be coupled to one or more host computer through the fabric interconnect. A device may have multiple paths through the fabric to the same host or host adapter port. Also, several upper layer protocol (ULP) devices may share the same connection to the fabric. For example, hard drive **210**A may be a drive array or JBOD including multiple hard drives or logical units coupled to the fabric through the same fabric connection. Note that the number and types of hosts and devices are for illustration purposes only, and the actual number and types of hosts and/or devices in a SAN may vary.

FIG. **3** illustrates a more detailed example of a storage network which includes a direct attached private loop **306** and a fabric **410**. Note that storage networks may be configured in a variety of different ways and many include one or more direct attach devices, SANs, and/or network attach (NAS) devices. Furthermore, note that fabrics and/or SANs are not limited to fibre channel technologies and architectures but may include various types of technologies. For example, some or all of a SAN may be based on the InfiniBand™ architecture or Small Computer System Interface over IP (iSCSI).

Host adapter **304** couples host system **402*a*** to private loop **306** and adapters **404*a*** and **404*b*** couple host system **402*b*** to fabric **410**. Note that host adapters **304**, **404*a*** and **404*b*** may be separate host bus adapter cards, for example. In other

embodiments, host adapters **304**, **404***a* and **404***b* may each refer to a separate host adapter port. Coupled to private loop **306** are one or more direct attach devices **308**. Direct attach device(s) are considered local to host system **402**A.

The host system **402***b* may be coupled to fabric **410** via host adapter **404***c*. Fabric **410** may include fibre channel switches **412** which are coupled to multiple fabric devices **408**. Each fibre channel switch **412** may connect to various fibre channel topologies such as point-to-point fibre channel connections or fibre channel loops. Each switch **412** may also connect to one or more other fibre channel switches. The fabric devices **408** may be various storage devices such as hard disk drives, optical drives, tape drives, etc. In some embodiments, fabric devices **408** may be any type of I/O device such as storage devices, printers, scanners, etc. as used in conjunction with computer systems.

Due to the large number of devices which may be present in fabric **410**, the time required for a host system **402** to discover and online all of the devices available through fabric **410** may be impractical. Furthermore, it may be unlikely that host system **402** actually needs to communicate with all of the various fabric devices **408**. The term "online" may be used herein to refer to a host creating a node as a representation in the host of a device and/or a communication mechanism or path to a device.

Within fabric **410**, a host or host adapter port may have multiple paths through the fabric to the same device. For example, FIG. **3** shows host system **402***a* having at least two paths to device **408***a*. One path is from **408***a* to switch **412***a* to host system **402***a*. Another path is from device **408***a* to switch **412***a* to switch **412***b* to host system **402***a*. In some embodiments, the same host adapter port may be able to reach a fabric device through several at least partially different paths within the fabric. In some embodiments, fabric **410** may include numerous switches and interconnect such that a host may have multiple paths to any number of devices within the fabric. Multiple paths may provide for redundancy in a host system's ability to communicate with a device.

FIG. **4** illustrates a host system **508** coupled to a fabric **510** according to one embodiment. The fabric may be implemented with one or more switches coupled to one or more storage devices or subsystems. Furthermore, the fabric may not be limited to fibre channel fabrics but may be extended to any type of switched storage network. A fabric driver **504** may provide an interface between the host system **508** and fabric **510**. A persistent repository **506** may be a data structure that stores information on the current configuration of the fabric devices. An administration application **502** may be a software program running on the host system **508** and accessible to a user (e.g., system administrator, other process, etc.). Administration application **502** may provide a mechanism for discovering fabric devices on-demand with user input thereby eliminating the need to discover all fabric devices accessible to a host system **508** at one time. As such, nodes may be created only for selected fabric devices (e.g., a subset of available fabric devices) during a discovery process. If a node is already created for a selected fabric device, it may not be necessary to re-create the node.

During a discovery process, the administration application **502** may query a fabric driver **504** for a list of devices visible to the host system **508**. In some embodiments, the query operations made by the administration application to obtain a list of the fabric devices may be made on a per host adapter port granularity or a set of host adapter ports granularity so that the number of devices returned by the query may be more manageable.

The fabric driver **504** may provide an interface for the host system **508** to the fabric **510**. The fabric driver **504** may be part of the operating system for the host system **508** and may include one or more modules for handling various functions required to interface the host system **508** to the fabric **510** such as protocol handling and device and/or transport layer operations. In one embodiment, the fabric driver **504** may be a Solaris kernel module or modules. The fabric driver **504** may provide the requested list of devices to the administration application **502**. A subset of these devices may then be selected through the administration application **502** and brought online by the fabric driver **504** so that the subset of devices are accessible from the host system **508**. Onlining the subset of devices may include the creation of a node within the operating system for each device wherein the node provides a reference for applications or other processes in the host system **508** to reference a corresponding device in the fabric. Thus, a node may provide a reference for an application or process running on the host system **508** to communicate with one of the fabric devices. When a fabric device is onlined, a communication path may be established between the host system **508** and the discovered fabric device. A discovered fabric device may have multiple paths through the fabric to a host **508**. Such a device may be referred to as a multi-path device. The host **508** may include a multi-path device manager, such as Traffic Manager from Sun Microsystems, Inc. Thus, it may be desirable to online a discovered fabric device for multiple communication paths to the host for management by a multi-path manager. The multi-path manager may provide for load balancing to devices over multiple paths in the fabric. The multi-path manager may provide for fail-over to another path in case a path to a multi-path device fails.

FIG. **5** illustrates a host system **800** coupled to a fabric **510** according to one embodiment. The host system **800** may include a fabric driver **504** for communicating with a fabric such as a fibre channel fabric. An interface **503** may be provided as an interface between the administration application **502** and the fabric driver **504**. In one embodiment, the interface **503** may be part of the operating system libraries and may be useable by other applications on the host system. In further embodiments, the interface **503** may be part of a particular application residing on the host system and useable by other applications, or implemented as part of fabric driver **504**.

The interface **503** may be provided to the fabric driver **504** that includes a device centric discovery interface **520** to represent a multi-path device as a single device while showing transport information for each path to the device. The device centric discovery interface **520** may allow a user (e.g., an application) to discover information about fabric devices from a device centric perspective. The device centric discovery interface **520** may return device centric discovery information such that a multi-path device is shown as a single device with transport information shown for each path to the device. In some embodiments, several upper layer protocol (ULP) devices may share the same connection to the fabric. The device centric discovery interface **520** may provide multi-path discovery information for each ULP device. For example, a SCSI drive array may include multiple SCSI LUN (Logical Unit Number) devices from the same SCSI target in the fabric. The device centric discovery interface **520** may return discovery information from the perspective of each SCSI LUN.

The interface **503** to the fabric driver may also include a transport centric discovery interface **522** to provide fabric device discovery information from a transport perspective.

The transport centric interface **522** may return transport information according to each path. Thus, the transport centric discovery interface **522** shows each path separately, even though two or more paths may be connected to the same device.

In one embodiment, including both a device centric discovery interface **520** and a transport centric discovery interface **522**, a user may discover fabric devices from a device centric perspective, a transport centric perspective, or both. For multi-path devices, the device centric discovery interface shows the device with transport information grouped together for each path to the device. Multi-path devices may be less easily identified using the transport centric discovery interface **522** since each path is shown separately and may not be grouped according to the device.

The interface **503** to the fabric driver may also include a device centric configuration interface **524** to configure multiple paths to fabric devices from a device perspective. For a particular device, the device centric configuration interface **524** may provide for configuring (e.g., onlining) multiple paths to the device. A user may desire to configure all discovered fabric connections to a certain device to take advantage of multi-path management support provided within the host. In one embodiment, for a specified device the device centric configuration interface **524** may configure each discovered path in the fabric to the specified device.

The interface **503** to the fabric driver may also include a transport centric configuration interface **526** to configure specific paths to fabric devices from a transport perspective. The transport centric configuration interface **526** may provide for configuring a specific path to a device.

Administration application **502** may provide a mechanism to select and online a subset of the visible fabric devices. The mechanism to select and online a subset of devices may utilize a device centric and/or transport centric interface to the fabric driver to discover and configure devices. In one embodiment, the administration application **502** may be run outside of the boot process so that device discovery and online operations do not increase the host system **508** boot time. The administration application **502** may be run on-demand so that fabric devices may be selected and brought online on demand. For one embodiment, a user may use the administration application **502** to request a list of fabric devices available through one or more host adapter I/O ports of the host system. The administration application **502** may include a command line interface or a graphical user interface (or both) for displaying the list to the user. Fabric device information may be requested from a device perspective, a transport perspective, or both. Through the administration application the user may then select a desired subset of the listed devices and may request that the selected devices be brought online. In one embodiment, the user may select a device from a device centric view and request that the selected device be brought online with associated multiple paths. In further embodiments, the user may select a device from a transport centric perspective and request that the selected device associated with the selected path be brought online.

For one embodiment, the fabric driver **504** may include various sub-modules. For example, in a fabric channel implementation, the fabric driver **504** may include a ULP device layer **802**. ULP device module(s) in layer **802** may be part of the operating system kernel and may perform all protocol related operations required for fabric channel use on the host operating system. For example, a ULP module may be a SCSI over fiber channel encapsulation driver module for supporting SCSI over fiber channel. The fabric driver may also include a transport layer **804**. This layer may include one or more modules that may perform generic fabric operations. In one embodiment, the transport layer may include a fiber channel port (FP) driver for each fibre channel port on the host system. Each FP driver may also be part of the operating system kernel. The FP driver may perform generic fiber channel operations such as topology discovery (e.g., loop, point-to-point, fabric, etc.), device discovery (on various topologies), handling extended link services, handling link state changes, etc. The fabric driver **504** may also include host adapter (HA) drivers **808** for each host adapter/controller board on the host system. For example, FCA drivers may be present for fiber channel adapters having fiber channel ports on the host system.

The interface **503** may provide application programming interfaces (APIs) for the administration application **502** to make queries in order to obtain a list of devices connected to one or more host adapter ports. Fabric device information may be returned according to a device centric perspective and/or a transport centric perspective. In one embodiment, interface **503** may interact with a device layer and/or a transport layer of the fabric driver **504** to execute the query irrespective of the interconnect topology of the host adapter ports. For example, the fabric driver **504** may obtain the list of devices connected to a fibre channel switched fabric by querying a fabric name server. The fabric name server may be located within a fabric switch or distributed across the fabric switches and may maintain information about the various fabric devices. The fabric name server may include a database of objects. Each fabric attached device may register or query useful information in the name server, including, name, address, class of service capability of other participants, etc. Fabric driver **504** may also obtain upper layer protocol (ULP) device information. The interface **503** or fabric driver **504** may also provide an API for obtaining a list of direct attach devices for a host. For example, the fabric driver **504** may obtain the loop map for a host system's private loop topology.

The following examples illustrate information that may be returned by transport centric discovery interface **522** and device centric discovery interface **520** from the transport and device layers of fabric driver **504**. In the first example below, transport centric information obtained through the transport centric interface, based on a port World Wide Name (WWN), is shown. As shown below, a device which may be connected to the fabric through two host adapter ports, 50020f2300006077 and 50020f2300006107, is presented separately for each port WWN.

Port WWN	Device Type	Capacity	Configured Info
Fabric FCA port c4			
50020f2300006077	SCSI-disk	36G	unconfigured
50020f2300005f24	SCSI-disk	36G	configured
210100e08b247c12	IP		configured
Fabric FCA port c5			
50020f2300006107	SCSI-disk	36G	unconfigured
50020f23000063a9	SCSI-disk	36G	configured
210100e08b245f12	IP		configured
Fabric FCA port c6			

In the next example, device centric information obtained through the transport centric interface based on an Upper Layer Protocol (ULP) device identifier, is shown. As shown in the example, a device which has ULP unique ID 60020F20000063A93AEFFB5C000D6984 is connected to both c4 and c5 host adapter ports and is presented as a single

ULP device. For FCP SCSI, the Global Unique ID may be used to indicate the ULP device. The multi-path (MP) support field below indicates if the fibre channel host adapter port is enabled with multi-path management support or not. The first three devices have the same FC connection since they are SCSI devices from the same SCSI target device.

FC connection	Configured	MP support	Status
FCP SCSI ID 60020F20000063A93AEFFB5C000D6984 (SCSI Disk Device Capacity: 36G)			
c4 50020f2300006077	no	yes	ok
c5 50020f2300006107	no	yes	ok
FCP SCSI ID 60020F20000063A93AEFFB3000098A19 (SCSI Disk Device Capacity: 36G)			
c4 50020f2300006077	no	yes	ok
c5 50020f2300006107	no	yes	ok
FCP SCSI ID 60020F20000063A93AEFFB200003D112 (SCSI Disk Device Capacity: 36G)			
c4 50020f2300006077	no	yes	ok
c5 50020f2300006107	no	yes	ok
FCP SCSI ID 60020F20000063A93AEFDE18000589DC (SCSI Disk Device Capacity: 36G)			
c4 50020f2300005f24	yes	yes	ok
c5 50020f23000063a9	yes	yes	ok
.			
.			
.			

Using the device centric and/or transport centric interfaces, a user may discover information about fabric devices accessible from a host and select devices for on-demand configuration including node creation. In one embodiment, for fabric topologies connected to the host system, operating system device nodes may not be created until an on-demand request is made by the administration application. Upon such a request from the administration application **502**, the interface **503** and fabric driver **504** may provide a list of devices visible through one or more fabric host adapter ports according to a device centric or a transport centric perspective. The list of fabric devices may be provided by the administration application **502** to a user, for example, through a graphical user interface. The user may use the administration application **502** to select and online particular devices which are desired to be used by the host system. The user may use administration application **502** to select a device and request a device centric configuration through interface **503** such that the selected device may be brought online with associated multiple paths. In other embodiments, devices may be configured from a transport centric perspective such that a device is configured for a selected device path.

For direct attach devices, e.g. private loop topologies, operating system device nodes may be created during driver attach (e.g., when the fabric driver is loaded during a reboot) for all devices visible through a direct attach port. In some embodiments, operating system device nodes may only be created for direct attach devices that support a particular protocol (e.g., Fibre Channel Protocol (FCP)/SCSI).

The user may also use administration application **502** to offline any devices which are no longer needed. Offlining may include removing access to a storage device from the host operating system by removing the node. The list of devices displayed to the user by the administration application **502** may include the information returned about the fabric devices from a device centric or transport centric perspective. The administration application **502** may request the fabric driver **504** to online or offline fabric devices as indicated by a user using the application's user interface. The administration application **502** may request the fabric driver **504** to online or offline fabric devices from a device centric or transport centric perspective.

In some embodiments, the administration application **502** may make requests to the fabric driver to obtain fabric device information and/or to online/offline fabric devices without the involvement of a user. For example, certain events or requests from other processes may trigger the administration application **502** to online a fabric device(s).

The persistent repository **506** may be stored in the host system, or in some central locale accessible to the host system **508** indicating the current fabric devices which have been onlined (e.g., devices for which an operating system device node has been created). The persistent repository **506** may store fabric device configuration information from a device centric or a transport centric perspective as described above. The information stored in the persistent repository **506** may be used so that the configuration of devices online for the host system **508** persists across reboots and shutdowns. For example, when the host system **508** is rebooted the persistent repository may be read to determine which devices were online before the reboot and the fabric driver may be requested to online these same devices again.

The persistent repository **506** may be dynamically updated to reflect the state of the fabric devices. For example, if a fabric device which is currently online for the host system is disabled on the fabric (for example, a hard drive fails or is removed from the fabric), the fabric driver may generate an event causing the persistent repository **506** to be updated to reflect that the device or a particular path configuration for the device is now offline. Similarly, if the same device later is restored on the fabric, the device or a particular path configuration for the device may be onlined again (e.g., in response to an event) for the host system **508** and the persistent repository **506** may be dynamically updated to reflect the onlined status.

Turning to FIG. **6**, a flowchart is provided illustrating a device discovery process according to one embodiment. The discovery process may be part of a host system reconfiguration boot process. Alternatively, if a non-reconfiguration boot is performed of the host system in which device discovery is not performed, the discovery process may run when a previously created device node is accessed in the host system. This discovery process may also be performed if the host system's host adapter link is lost (e.g., cable pulled out).

During the discovery process a fabric driver is loaded as indicated at **602**. Note that if the fabric driver is already loaded, then it may not be necessary to load the fabric driver again. The link state of each host adapter port may be checked as indicated at **604** and **606**. If a port's link is down, the link may be designated as offline as indicated at **608**, and the discovery process for that link may wait for a state change in the link to online, as indicated at **610**. If the link is later restored, the discovery process may continue for that link.

If the link is up, the link type is determined, as indicated at **612**. In one embodiment, a fabric login is attempted through the link. If the fabric login is successful, the link is determined to be a fabric link. If unsuccessful the link is determined to be a direct attach link. For a direct attach link, one or more direct attach devices are discovered and brought online by creating operating system device nodes for each

direct attach device. If the link is a fabric link, the link is designated as such, but no device discovery or onlining of devices is performed for the fabric link as part of this discovery process, as indicated at **616**. This discovery process may be repeated any time a link goes out (e.g., cable pulled, power off, host reboot, etc.).

The device discovery process may provide for the discovery and onlining of direct attach devices since the discovery and onlining of such devices may be completed quickly due to the limited number of such devices. However, in some embodiments, for a host system's fabric links, device discovery may not be performed as part of the normal discovery process at reconfiguration boot up or the first time a node is attempted to be accessed after a link has been down and brought back up. Instead, fabric devices may be discovered using the on-demand node creation process described herein.

One embodiment of the on-demand node creation process is illustrated by the flowchart of FIG. **7**. A request may be received to discover fabric attached devices and associated paths, as indicated at **610**. The request may be a request to an interface to discover fabric devices from a device and/or transport centric perspective. In response to the request to discover fabric attached devices, a fabric may be requested to identify fabric devices and associated paths available to a host system, as indicated at **620**. A list of identified fabric devices and associated paths may be received from the fabric, as indicated at **630**. Note that the term "list" simply refers to the information or data provided by the fabric driver.

This list may be provided to a user for selection of a subset of the fabric devices, as indicated at **640**. Discovery information may be returned in a device centric perspective such that multi-path devices are each presented as a single device with transport information shown for each path to the device. In one embodiment, device centric discovery may be presented by grouping path information together for each device so that each device is presented with all of its discovered paths. Discovery information may alternatively or additionally be returned in a transport centric perspective such that device transport information for each port is listed. In one embodiment, transport centric discovery may be presented by grouping device transport information together for each port so that each discovered device path from a host adapter port is presented together for the port. Alternatively, selection of a subset of the fabric devices may be performed without user involvement, as indicated at **645**.

A request may be received to on-line the subset of identified fabric devices, as indicated at **650**. In one embodiment, the user may select a device from a device centric view and request that the selected device be brought online for each path presented for the device. Thus, a user may select a device for device centric configuration to online the selected device for all of the paths presented for the device. In another embodiment, the user may select a device from a transport centric view and request that the selected device be brought online for the selected path. In response to the request to on-line the subset of identified fabric devices, a node or nodes may be created for each fabric device of the subset not already on-line, as indicated at **660**. A node provides a mechanism for processes to communicate with the corresponding device from the host system. The on-line status on each fabric device of the subset may be stored, as indicated at **670**.

Another embodiment of the on-demand node creation process is illustrated by the flowchart of FIG. **8**. An administration application may make a request for a list of fabric devices, as indicated at **702**. This request may have been initiated by a user using the administration application for the host system. In other embodiments, this process may be initiated automatically, for example in response to an event (e.g., fibre channel protocol event) or a request from another application or process. The request may be made to a device centric or transport centric interface, as described above, to request fabric device information in a device centric or transport centric perspective. The device centric and/or transport centric interface may then request fabric device information from a fabric driver, as indicated at **704**. The fabric driver may access a fabric name server to obtain the requested information and return the requested list of fabric devices to the administration application. Note that the term "list" simply refers to the information or data provided by the fabric driver, not any particular format.

The administration application may receive the list and a subset of devices may be selected from the list, as indicated at **706**. If a device centric discovery request was made, discovery information may be returned in a device centric perspective such that multi-path devices are each presented as a single device with transport information shown for each path to the device. If a device centric discovery request was made, discovery information may be returned in a transport centric perspective such that device transport information is listed for each host adapter port. In one embodiment, the list may be displayed through a graphical user interface to a user. In other embodiments, a command line or textual user interface may be used. The user may select particular devices from the list to be onlined or offlined. The user may select a device in a device centric view such that each discovered path for the device is selected. In some embodiments, a user may select devices to be onlined for particular paths from a transport centric view. The administration application may then call the fabric driver to online or offline devices as indicated by the selections made for the subset from the list, as indicated at **708**. The driver may then attempt to online the devices selected to be onlined (and may offline the devices selected to be offlined), as indicated at **710**. Onlining a device may include the creation of an operating system device node for that device. The device node may provide a mechanism for processes to communicate with the corresponding device from the host system.

A persistent repository may be updated (or created if not already existing) to indicate which devices and associated paths are currently online including device centric and/or transport centric data, as indicated at **712**. During operation, the persistent repository may be updated to reflect any changes in fabric devices. For example, if a device and/or associated path indicated by the persistent repository to be currently onlined is removed from the fabric, the persistent repository may be updated accordingly as indicated at **714**. Changes in the fabric may be detected by an event generated in the fabric and communicated to the administration application (or library) by the fabric driver.

Turning to FIG. **9**, a flowchart shows a method for onlining selected fabric devices on-demand from a list of fabric devices according to one embodiment. In one embodiment, a list of fabric devices may have been obtained by the method illustrated in FIG. **7** or **8**. A user may choose a subset of devices from this list to online and pass this list of devices to the administration application, as indicated at **932**. The administration application may then call an interface to the fabric driver to online the selected devices. For example, the administration application may call a fabric driver (via an interface) to online the selected devices, as

indicated at **934**. In one embodiment, the user may have selected a device from a device centric view and requested that the selected device be brought online with associated multiple paths. In further embodiments, the user may have selected a device from a transport centric view and requested that the selected device associated with the selected path be brought online. The fabric driver may attempt to online each of the selected devices in the list and report each successful online to the interface, as indicated at **936**. For every device successfully onlined by the fabric driver, the administration application updates a persistent repository to reflect devices currently onlined, as indicated at **938**. Note also that offlining a node through the administration application may cause an event to be generated to trigger an update the persistent repository.

Turning to FIG. **10**, one embodiment is illustrated of a mechanism for dynamically updating the persistent repository to reflect changes that occur in the fabric. A device state change may occur in the fabric, as indicated at **950**. For example, a fabric disk drive online for the host system may be removed from the fabric. An event may be generated to indicate this change. The fabric driver may inform the interface/application of such changes that affect the device list, as indicated at **952**. The administration application may update the contents of the persistent repository in response to receiving notification of a change, as indicated at **954**, so that the persistent repository is dynamically updated to reflect such changes. The persistent repository may be stored, for example, on persistent storage such as a disk drive or non-volatile memory.

The persistent repository may allow a host computer's fabric device configuration to persist across reboots and shutdowns as illustrated in FIG. **11** for one embodiment. On a host reboot (**970**), a component of administration application (e.g., in the Solaris 'rc' scripts) may read the persistent repository to determine which devices were previously online, as indicated at **972**. The administration application then calls the fabric driver (via the interface) to online the fabric devices that were onlined prior to the reboot, as indicated at **974**. The indicated at **976**. The library may fabric driver attempts to online each device indicated by the persistent repository and reports successful onlines to through the interface to the application, as then update the persistent repository to reflect the currently online devices, as indicated at **978**.

An example of an entry in a persistent repository is: /devices/pci@1f,4000/pci@2/SUNW,qlc@4/fp@0,0:fc::50020f23000000e6. In this example, the entry identifies the /devices path to an onlined FC port. In other embodiments, additional information may be included. For example, the repository may identify onlined fabric devices on a per LUN basis. The persistent repository may store device configuration information in a device centric perspective, transport centric perspective, or both.

Note that the flow charts described herein represent exemplary embodiments of methods. The methods may be implemented in software, hardware, or a combination thereof. The order of method may be changed, and various elements may be added, reordered, combined, omitted, modified, etc. For example in FIG. **7**, the persistent repository may store the online status (**670**) before, after or during creation of a node (**660**).

Various modifications and changes may be made as would be obvious to a person skilled in the art having the benefit of this disclosure. Note also that the flow charts described herein do not necessary require a temporal order. It is

intended that the following claims be interpreted to embrace all such modifications and changes and, accordingly, the specifications and drawings are to be regarded in an illustrative rather than a restrictive sense.

Various embodiments may further include receiving, sending or storing instructions and/or data implemented in accordance with the foregoing description upon a computer readable medium. Generally speaking, a computer readable medium may include storage media or memory media such as magnetic or optical media, e.g., disk or CD-ROM, volatile or non-volatile media such as RAM (e.g. SDRAM, DDR SDRAM, RDRAM, SRAM, etc.), ROM, etc. as well as transmission media or signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as network and/or a wireless link.

What is claimed is:

1. A system, comprising:

one or more host adapter ports for coupling the system to a fabric network, wherein one or more devices attached to the fabric network are visible to the system through the one or more host adapter port;

one or more processors configured to execute:

a fabric driver configured to interface the system to the fabric network through the host adapter ports; and

a device centric discovery interface configured to provide an interface to the fabric driver to obtain information about the devices in the fabric network, wherein the device centric discovery interface is configured to return device centric discovery information such that a multi-path fabric device is presented as a single device with transport information provided for each path to the multi-path device.

2. The system as recited in claim **1**, wherein the processor is further configured to execute a device centric configuration interface configured to provide an interface to the fabric driver for device centric configuration of the devices in the fabric for use by the system such that a requested fabric device is configured for use by the system on multiple paths in the fabric network.

3. The system as recited in claim **2**, wherein the processor is further configured to execute an application configured to request the device centric discovery interface to provide a list of fabric devices attached to the fabric that are visible to the system through one of the adapter ports;

wherein the device centric discovery interface is further configured to provide a device centric presentation of fabric devices to the application in response to the request from the application;

wherein the application is further configured to indicate to the device centric configuration interface a selected subset of the fabric devices from the presented fabric devices to be brought online for access from the host system; and

wherein the device centric configuration interface is further configured to online through the fabric driver the selected subset of fabric devices so that each of the selected subset of fabric devices is accessible from the host system on all of its presented fabric paths.

4. The host system as recited in claim **3**, wherein the application is further configured to:

display the list of fabric devices to a user through a user interface in a device centric presentation such that each fabric device is shown grouped with its all its discovered fabric paths; and

provide through the user interface for the user to select devices from the list as the selected subset of the fabric device to be brought online for all their respective fabric paths.

5. The system as recited in claim 2, wherein said in response to a request from the device centric configuration interface, the fabric driver is further configured to create device nodes within the host system for each requested fabric device, wherein each device node provides a mechanism for accessing a corresponding one of the subset of fabric devices through an operating system executing on the processor.

6. The system as recited in claim 1, wherein the processor is further configured to execute a transport centric discovery interface configured to provide an interface to the fabric driver to obtain information about the devices in the fabric network, wherein the transport centric discovery interface is configured to return transport centric discovery information such that a multi-path device is presented as a separate device for each path to the multi-path device.

7. The system as recited in claim 1, wherein the processor is further configured to execute a transport centric configuration interface configured to provide an interface to the fabric driver for transport centric configuration of the devices in the fabric for use by the system such that fabric device is configured for use by the system on a requested fabric transport path.

8. The system as recited in claim 1, wherein the one or more host adapter ports comprise a Fibre Channel host adapter port.

9. The system as recited in claim 1, wherein the device centric discovery interface is comprised within a library, wherein the library further comprises a persistent repository interface configured to update a persistent repository for each fabric device successfully brought online for the host system to indicate which devices are currently online.

10. The system as recited in claim 9, wherein the system is configured to, in response to a reboot of the host system:

read the persistent repository; and

request the fabric driver to online the devices indicated by the persistent repository to have been onlined prior to the reboot.

11. The system as recited in claim 1, wherein the fabric network comprises a Fibre Channel switched fabric comprising a plurality of Fibre Channel switches.

12. The system as recited in claim 1, wherein the fabric network is part of a storage area network (SAN), and wherein the fabric devices comprise storage devices.

13. A method, comprising:

discovering a plurality of devices on a fabric network;

displaying information describing the plurality of discovered devices in a device-centric format, wherein in the device-centric format each device is displayed as a single device with one or more associated paths each corresponding to a particular transport connection to a host, wherein at least one of the discovered devices is displayed in the device-centric format as a single device grouped with transport information for multiple different connections to the host; and

configuring a selected one of the devices for use by the host on one or more associated paths in response to user input.

14. The method as recited in claim 13, wherein said configuring comprises a device centric configuration of the selected device in the fabric for use by the system such that the selected device is configured for use by the system on multiple paths in the fabric network during a particular configuration.

15. The method as recited in claim 13, wherein said configuring comprises creating a device node within the host for each associated path, wherein each device node provides a mechanism for accessing through an operating system executing on the host the corresponding fabric device on a particular associated path in the fabric network.

16. The method as recited in claim 13, further comprising displaying information describing the plurality of discovered devices in a transport-centric format, wherein in the transport-centric format a multi-path device is presented as a separate device for each path to the multi-path device.

17. The method as recited in claim 13, further comprising updating a persistent repository for each fabric device successfully configured for the host system to indicate which devices are currently online.

18. The method as recited in claim 17, further comprising, in response to a reboot of the host:

reading the persistent repository; and

configuring the devices indicated by the persistent repository to have been onlined prior to the reboot for use by the host.

19. The method as recited in claim 13, wherein the fabric network comprises a Fibre Channel switched fabric comprising a plurality of Fibre Channel switches.

20. The method as recited in claim 13, wherein the fabric network is part of a storage area network (SAN), and wherein the fabric devices comprise storage devices.

21. A storage area network comprising:

a host system;

a plurality of fabric devices coupled to the host system via a fabric network;

wherein the host system is configured to:

discover one or more of the plurality of fabric devices on the fabric network;

display information describing the one or more discovered fabric devices in a device-centric format, wherein in the device-centric format each device is displayed as a single device with one or more associated paths each corresponding to a particular transport connection to the host system, wherein at least one of the fabric devices is displayed in the device-centric format as a single device grouped with transport information for multiple different connections to the host system; and

configure on demand a selected one of the devices for use by the host on one or more associated paths in the fabric network.

22. A host system, comprising:

one or more adapter ports for connecting to a fabric network;

a fabric driver configured to interface the host system to the fabric network;

wherein the host system is configured to:

discover one or more of a plurality of devices coupled to the fabric network;

display information describing the one or more discovered devices in a device-centric format, wherein in the device-centric format each device is displayed as a single device with one or more associated paths each corresponding to a particular adapter port of the host system, wherein at least one of the discovered devices is displayed in the device-centric format as a single device grouped with transport information for multiple different connections to the host system; and

configure on demand a selected one of the devices for use by the host on one or more associated paths in the fabric network.

* * * * *

EXHIBIT B

US006151683A

United States Patent [19]

Wookey

[11] **Patent Number: 6,151,683**

[45] **Date of Patent: Nov. 21, 2000**

[54] **REBUILDING COMPUTER STATES REMOTELY**

[75] Inventor: **Michael J. Wookey**, Sunnyvale, Calif.

[73] Assignee: **Sun Microsystems, Inc.**, Palo Alto, Calif.

[21] Appl. No.: **08/829,276**

[22] Filed: **Mar. 31, 1997**

[51] **Int. Cl.**[7] **G06F 11/00**

[52] **U.S. Cl.** **714/2**; 714/15; 714/25; 714/26

[58] **Field of Search** 395/182.02, 182.13, 395/182.19, 185.02, 185.03; 714/2, 15, 25, 11, 26; 370/221, 258, 222, 224, 452, 453; 707/501; 706/50, 59, 45

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,567,560	1/1986	Polis et al.	364/184
4,637,013	1/1987	Nakamura	370/221
4,709,365	11/1987	Beale et al.	395/182.02
5,101,402	3/1992	Chiu et al.	709/224
5,155,847	10/1992	Kirouac et al.	395/200.51
5,299,312	3/1994	Rocco, Jr.	395/182.02
5,307,354	4/1994	Cramer et al.	395/182.02
5,400,246	3/1995	Wilson et al.	364/146
5,471,399	11/1995	Tanaka et al.	364/491
5,487,169	1/1996	Vraney et al.	395/700
5,491,791	2/1996	Glowny et al.	395/183.13
5,495,610	2/1996	Shing et al.	395/200.51
5,539,869	7/1996	Spoto et al.	345/336
5,600,796	2/1997	Okamura et al.	395/181
5,655,081	8/1997	Bonnell et al.	395/200.32
5,668,944	9/1997	Berry	395/184.01
5,696,486	12/1997	Poliquin et al.	395/200.54
5,699,505	12/1997	Srinivasan	714/10
5,726,912	3/1998	Krall, Jr. et al.	702/186
5,727,144	3/1998	Brady et al.	395/182.04
5,751,964	5/1998	Ordanic et al.	395/500.54
5,758,071	5/1998	Burgess et al.	395/200.5
5,825,944	10/1998	Wang	382/309
5,908,471	6/1999	Lach et al.	714/805
5,909,540	6/1999	Carter et al.	714/4
5,944,839	8/1999	Isenberg	714/26

OTHER PUBLICATIONS

"Remote Systems Diagnostics Installation & User Guide, Remote Systems Monitoring (SunReMon™), Remote Dial–in Analysis (SunRDA ™)," Release 1.0.1, Sun Microsystems, Mountain View, California, Nov. 1996, (116 pages).

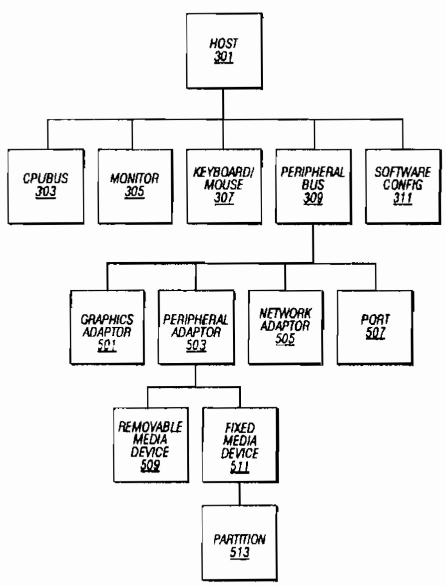
"Solstice™ SyMON™ User's Guide," Revision A, Sun Microsystem Computer Company, Mountain View, California, May 1996 (116 pages).

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Pierre Eddy Elisca
Attorney, Agent, or Firm—Skjerven Morill MacPherson, LLP

[57] **ABSTRACT**

A representation of the state of a computer, based on diagnostic data of the computer, is built by extracting system information from the diagnostic data and building a component based representation of the computer using the extracted system information. A static tree definition of a computer system is provided which is formed by element types in a fixed hierarchical relationship. A plurality of token types are provided, each of the token types being associated with one of the element types. The token types are component based data types. Respective segments of the incoming data that are defined by respective token types are identified and stored as tokens in a token data base. Each of the tokens has a value field holding a value associated with the element and a parent field referring to an element with which the token is associated. For each element in the static definition, the token data base is searched for associated tokens and a host state is built based on the static state definition and the extracted associated tokens, the elements of the static state definition being given value by their associated tokens.

29 Claims, 15 Drawing Sheets



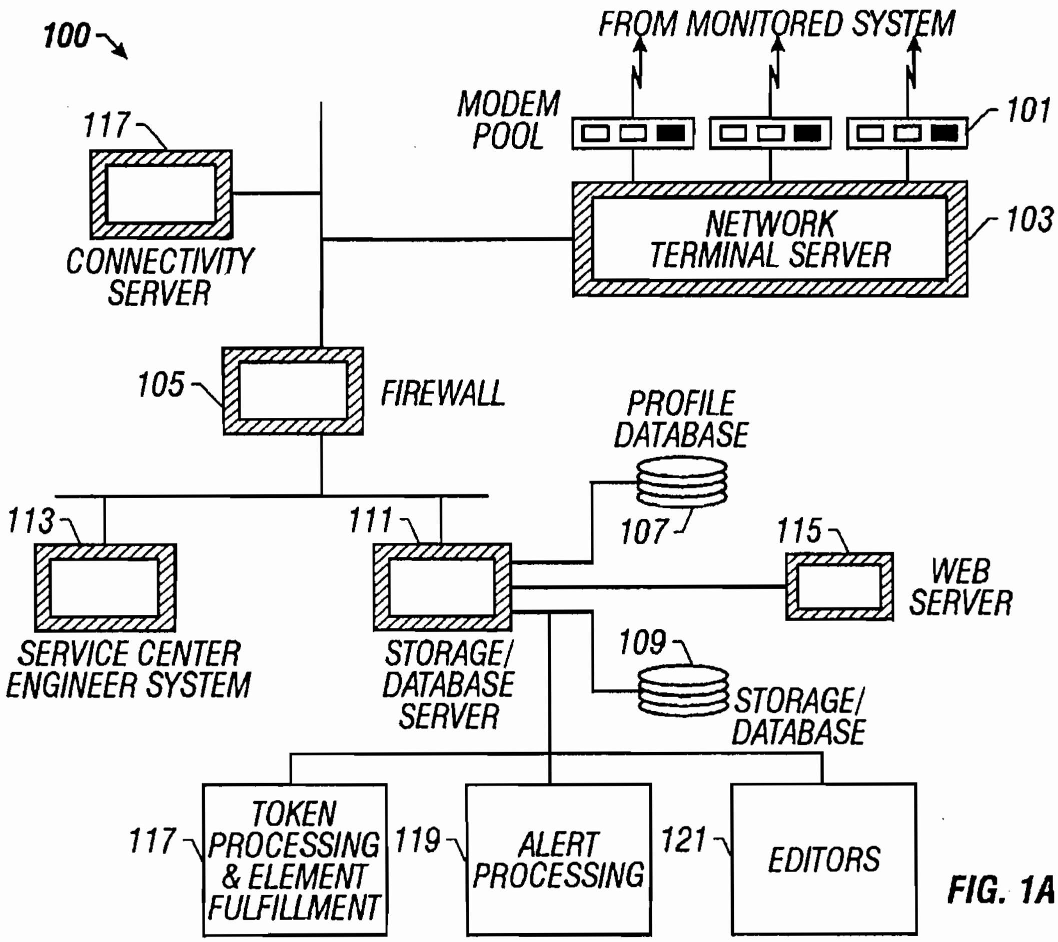

100
FROM MONITORED SYSTEM
MODEM POOL
101
117
CONNECTIVITY SERVER
NETWORK TERMINAL SERVER
103
105
FIREWALL
PROFILE DATABASE
107
113
111
115
WEB SERVER
SERVICE CENTER ENGINEER SYSTEM
STORAGE/ DATABASE SERVER
109
STORAGE/ DATABASE
117
TOKEN PROCESSING & ELEMENT FULFILLMENT
119
ALERT PROCESSING
121
EDITORS

FIG. 1A

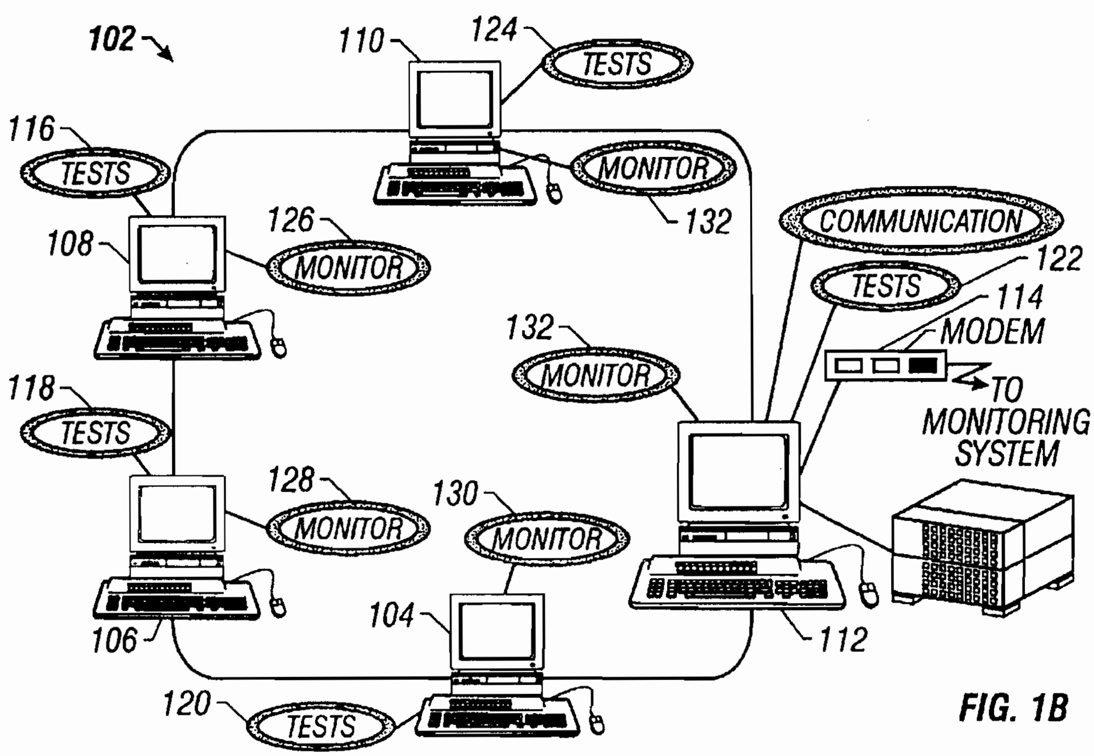

102
110
124
TESTS
116
TESTS
MONITOR
132
108
126
MONITOR
COMMUNICATION
TESTS
122
114
MODEM
132
MONITOR
TO MONITORING SYSTEM
118
TESTS
128
MONITOR
130
MONITOR
104
112
106
120
TESTS

FIG. 1B

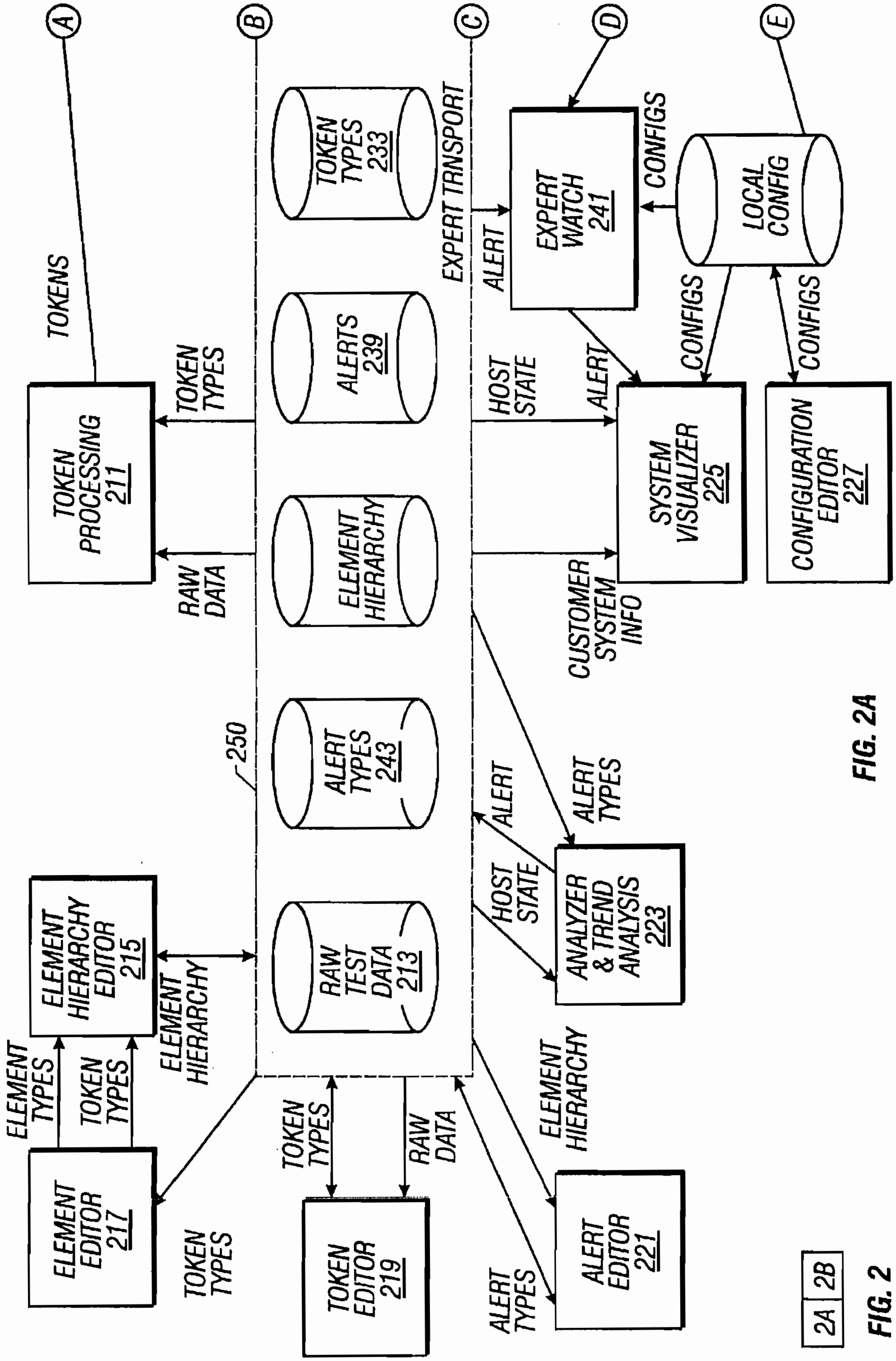

A
B
C
D
E
TOKENS
TOKEN TYPES
RAW DATA
TOKEN PROCESSING 211
TOKEN TYPES 233
ALERTS 239
ELEMENT HIERARCHY
ALERT TYPES 243
RAW TEST DATA 213
250
EXPERT TRNSPORT
ALERT
EXPERT WATCH 241
CONFIGS
LOCAL CONFIG
CONFIGS
CONFIGS
HOST STATE
ALERT
SYSTEM VISUALIZER 225
CONFIGURATION EDITOR 227
CUSTOMER SYSTEM INFO
ALERT
ALERT TYPES
HOST STATE
ANALYZER & TREND ANALYSIS 223
ELEMENT HIERARCHY EDITOR 215
ELEMENT TYPES
TOKEN TYPES
ELEMENT HIERARCHY
ELEMENT EDITOR 217
TOKEN TYPES
TOKEN TYPES
RAW DATA
TOKEN EDITOR 219
ALERT TYPES
ELEMENT HIERARCHY
ALERT EDITOR 221
2A 2B

FIG. 2A

FIG. 2

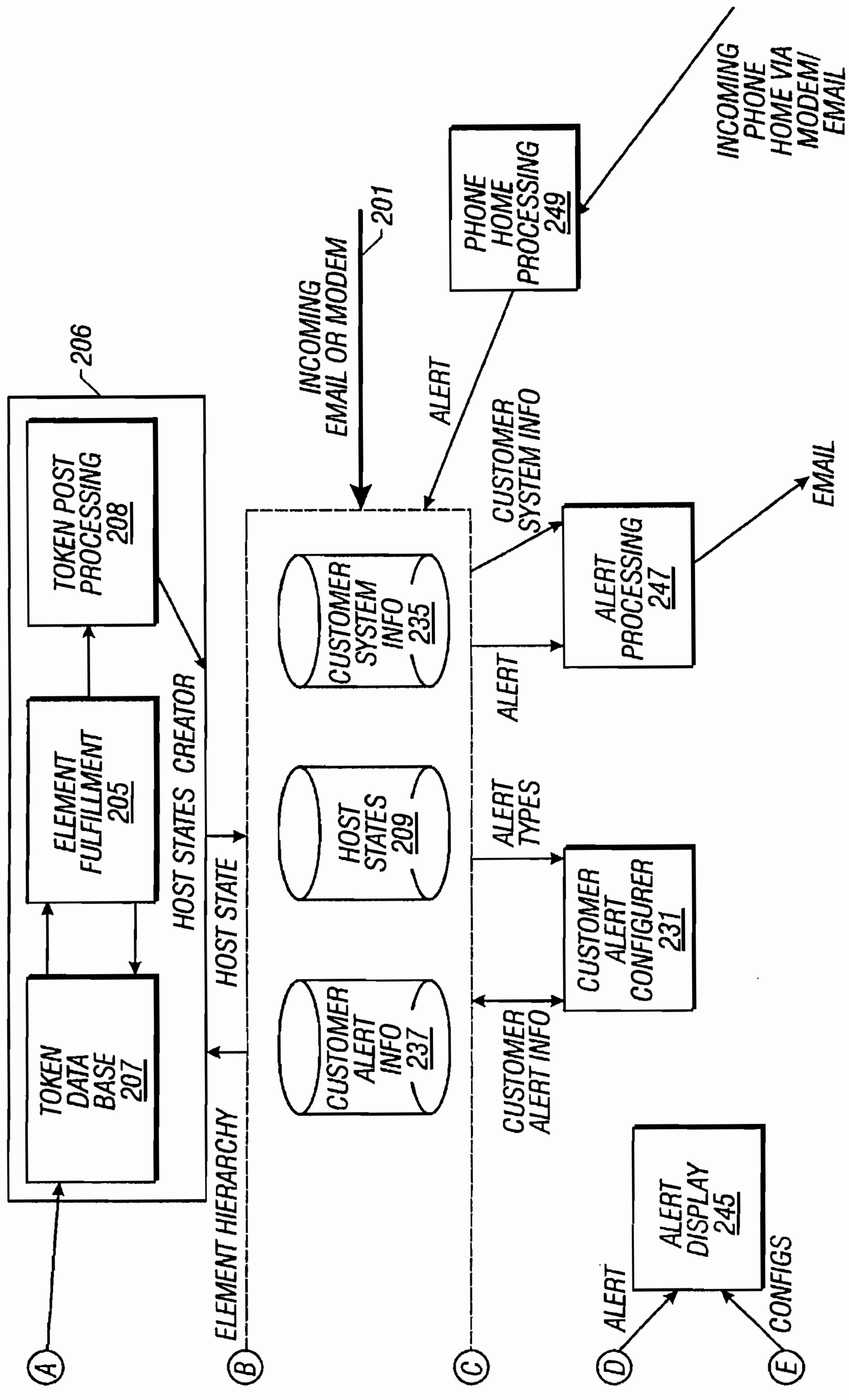

206
TOKEN POST PROCESSING 208
ELEMENT FULFILLMENT 205
TOKEN DATA BASE 207
HOST STATES CREATOR
HOST STATE
ELEMENT HIERARCHY
INCOMING EMAIL OR MODEM
201
PHONE HOME PROCESSING 249
INCOMING PHONE HOME VIA MODEM/ EMAIL
ALERT
CUSTOMER SYSTEM INFO 235
HOST STATES 209
CUSTOMER ALERT INFO 237
CUSTOMER SYSTEM INFO
ALERT
ALERT TYPES
CUSTOMER ALERT INFO
ALERT PROCESSING 247
EMAIL
CUSTOMER ALERT CONFIGURER 231
ALERT DISPLAY 245
ALERT
CONFIGS
A
B
C
D
E

FIG. 2B

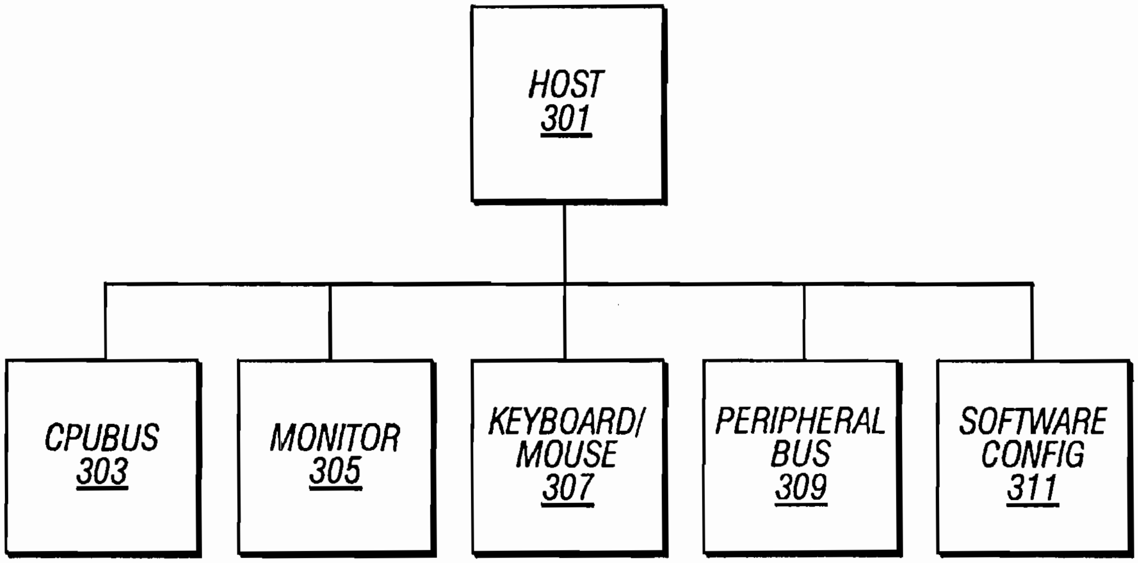

HOST
301
CPUBUS
303
MONITOR
305
KEYBOARD/
MOUSE
307
PERIPHERAL
BUS
309
SOFTWARE
CONFIG
311

FIG. 3

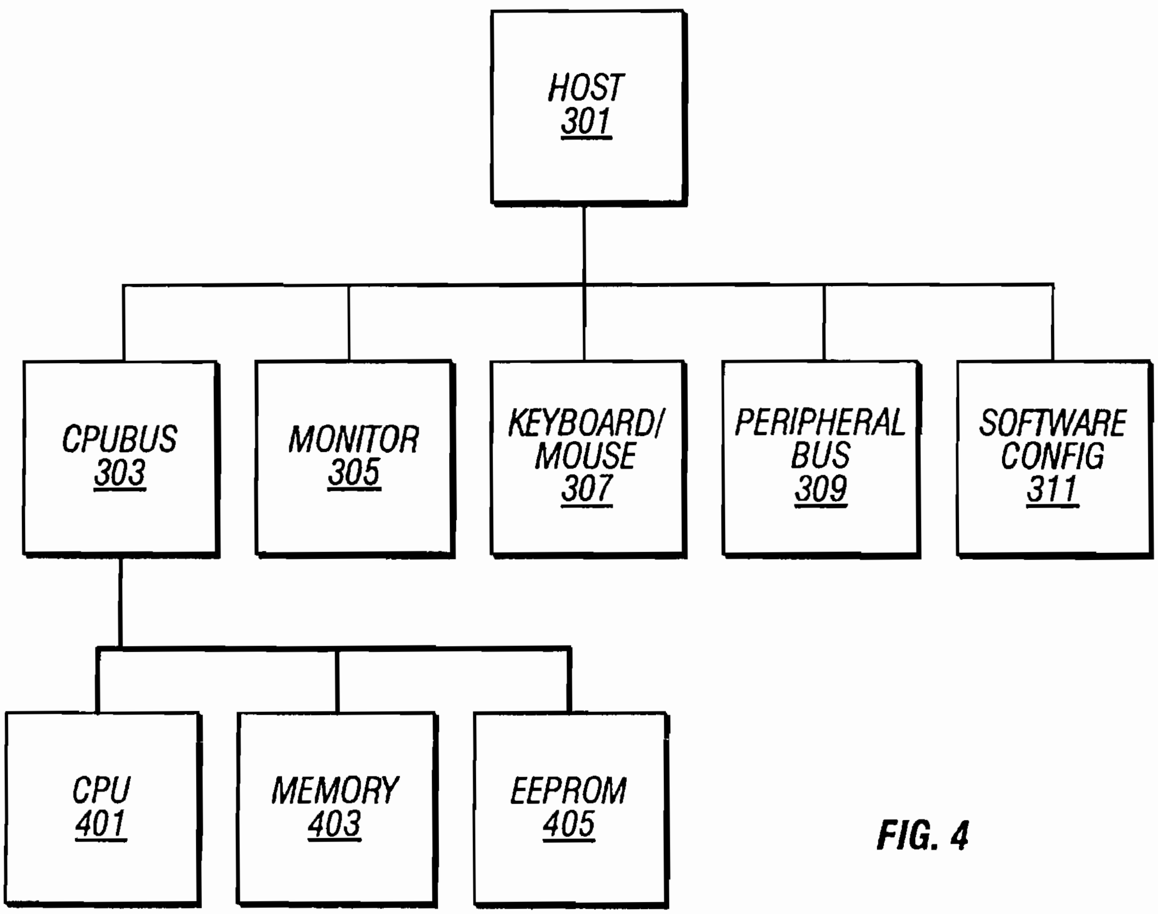

HOST
301
CPUBUS
303
MONITOR
305
KEYBOARD/
MOUSE
307
PERIPHERAL
BUS
309
SOFTWARE
CONFIG
311
CPU
401
MEMORY
403
EEPROM
405

FIG. 4

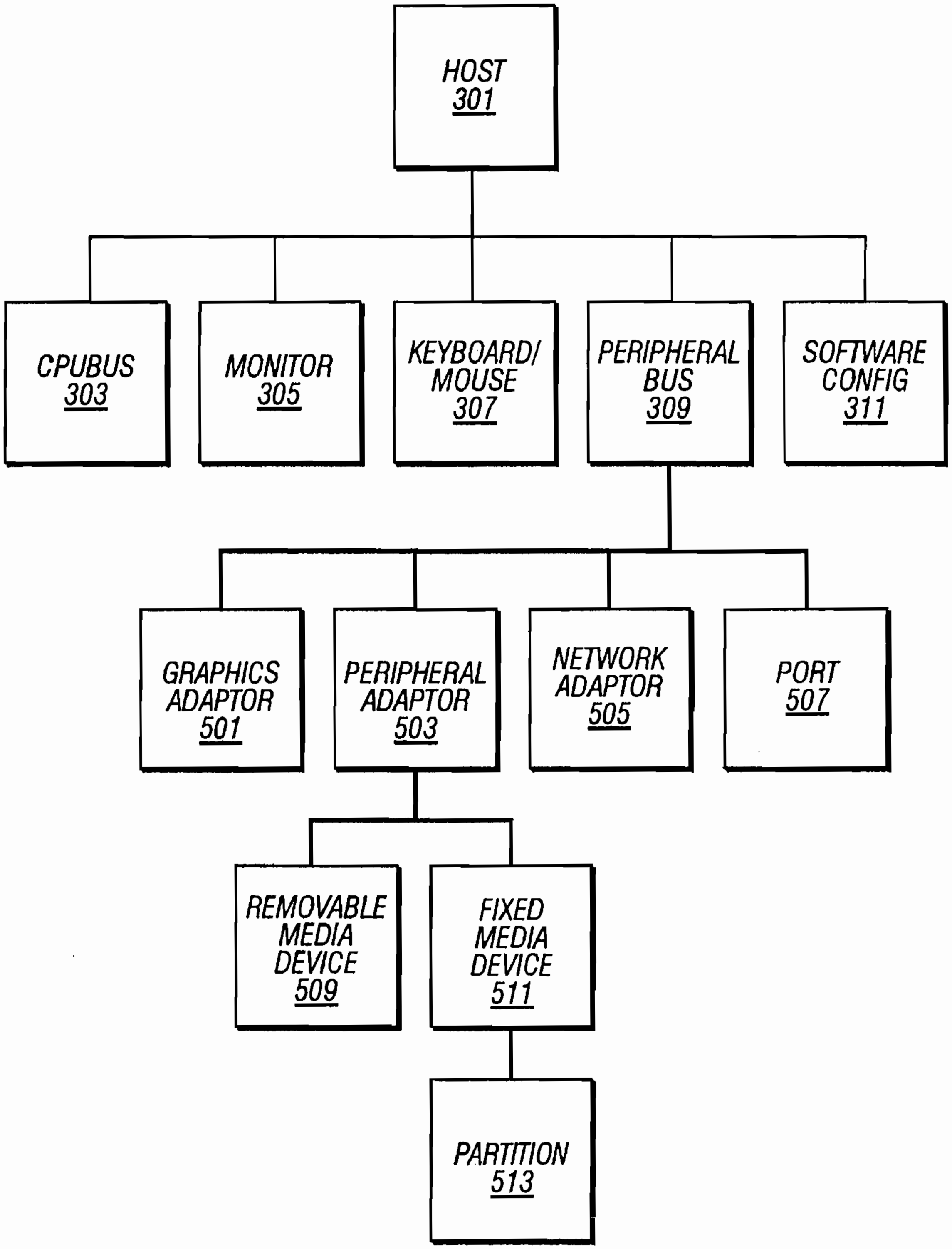

HOST 301
CPUBUS 303
MONITOR 305
KEYBOARD/ MOUSE 307
PERIPHERAL BUS 309
SOFTWARE CONFIG 311
GRAPHICS ADAPTOR 501
PERIPHERAL ADAPTOR 503
NETWORK ADAPTOR 505
PORT 507
REMOVABLE MEDIA DEVICE 509
FIXED MEDIA DEVICE 511
PARTITION 513

FIG. 5

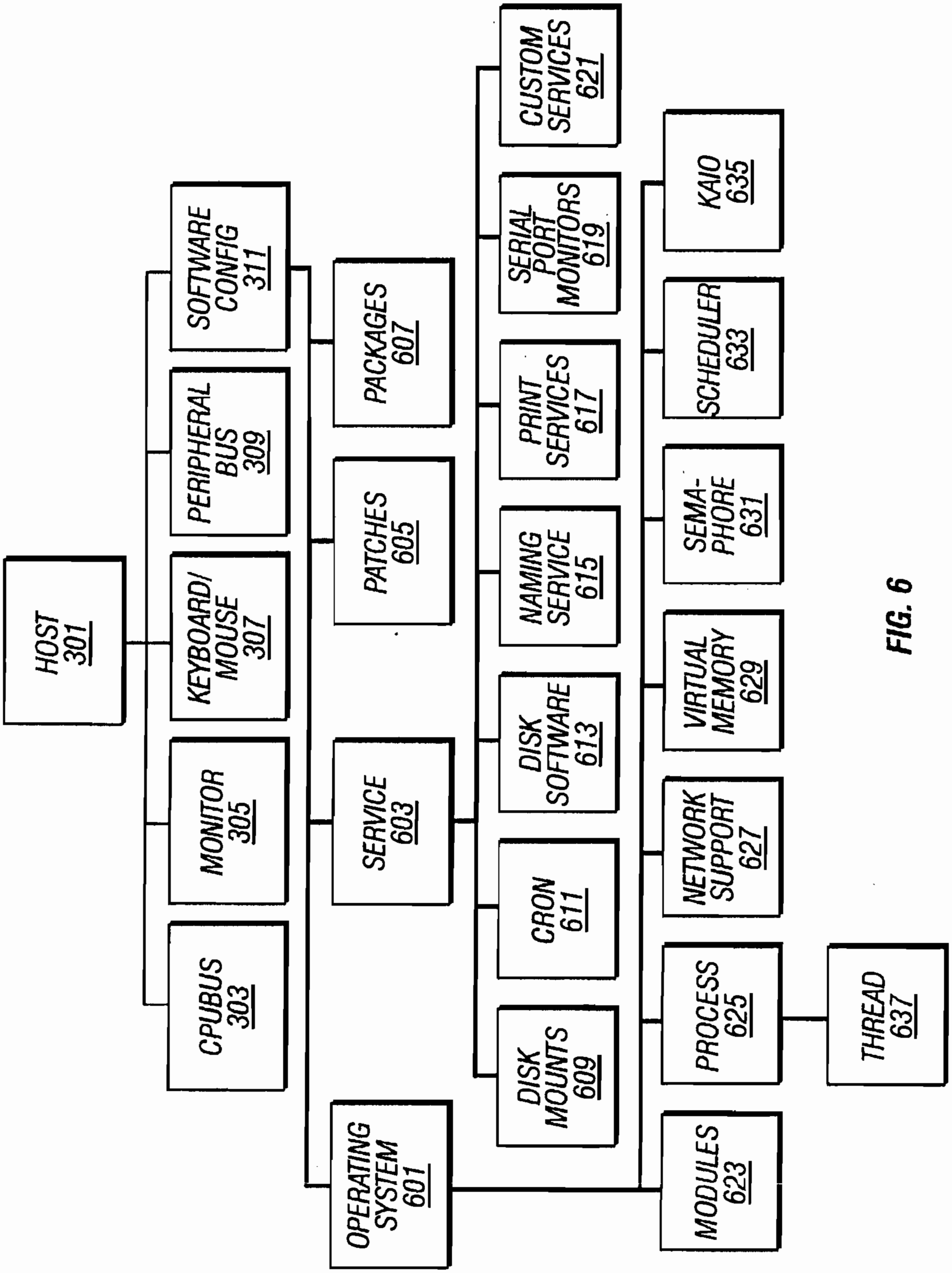

HOST 301
CPUBUS 303
MONITOR 305
KEYBOARD/ MOUSE 307
PERIPHERAL BUS 309
SOFTWARE CONFIG 311
OPERATING SYSTEM 601
SERVICE 603
PATCHES 605
PACKAGES 607
DISK MOUNTS 609
CRON 611
DISK SOFTWARE 613
NAMING SERVICE 615
PRINT SERVICES 617
SERIAL PORT MONITORS 619
CUSTOM SERVICES 621
MODULES 623
PROCESS 625
NETWORK SUPPORT 627
VIRTUAL MEMORY 629
SEMA- PHORE 631
SCHEDULER 633
KAIO 635
THREAD 637

FIG. 6

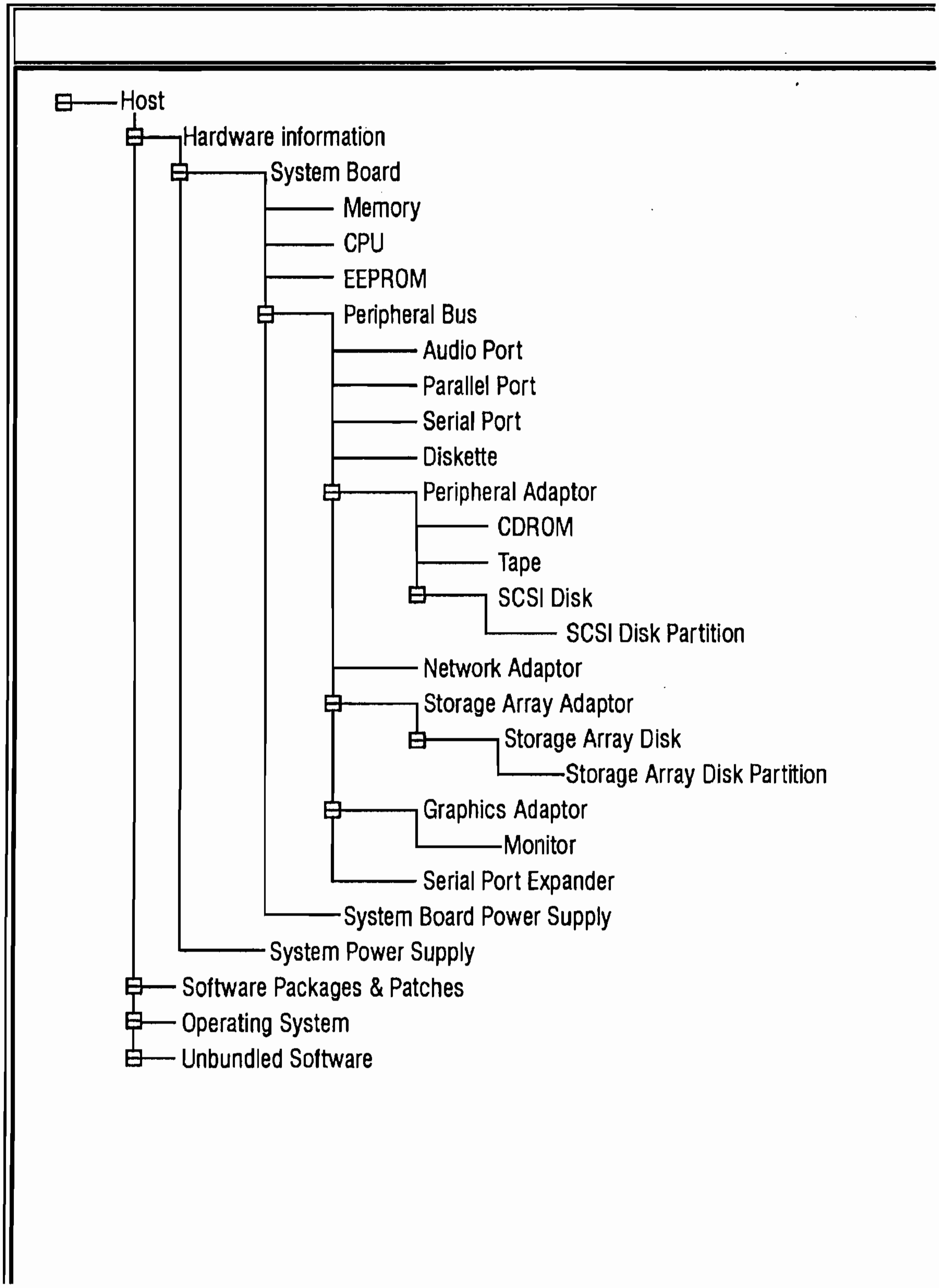

Host
Hardware information
System Board
Memory
CPU
EEPROM
Peripheral Bus
Audio Port
Parallel Port
Serial Port
Diskette
Peripheral Adaptor
CDROM
Tape
SCSI Disk
SCSI Disk Partition
Network Adaptor
Storage Array Adaptor
Storage Array Disk
Storage Array Disk Partition
Graphics Adaptor
Monitor
Serial Port Expander
System Board Power Supply
System Power Supply
Software Packages & Patches
Operating System
Unbundled Software

FIG. 7A

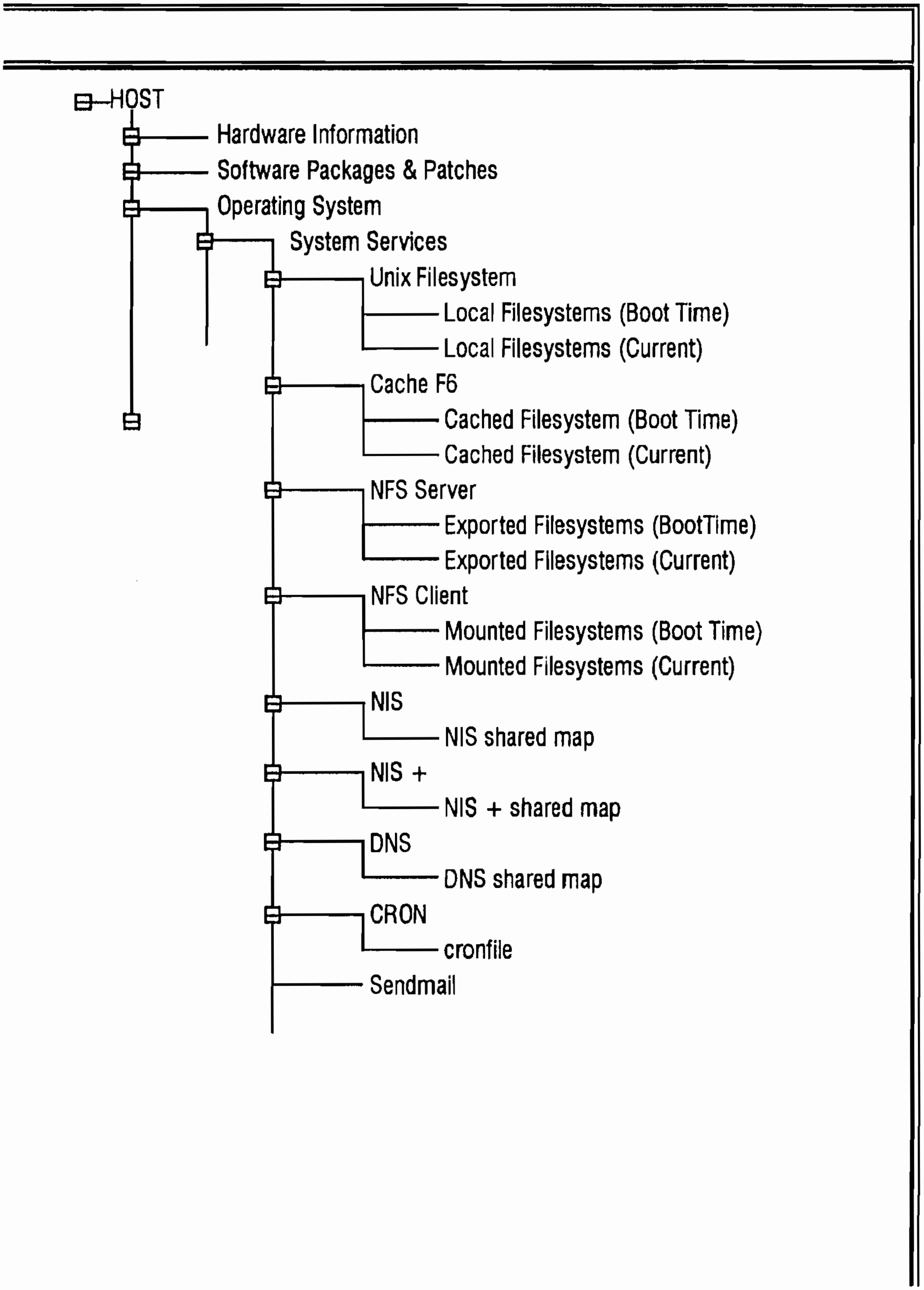

HOST
Hardware Information
Software Packages & Patches
Operating System
System Services
Unix Filesystem
Local Filesystems (Boot Time)
Local Filesystems (Current)
Cache F6
Cached Filesystem (Boot Time)
Cached Filesystem (Current)
NFS Server
Exported Filesystems (BootTime)
Exported Filesystems (Current)
NFS Client
Mounted Filesystems (Boot Time)
Mounted Filesystems (Current)
NIS
NIS shared map
NIS +
NIS + shared map
DNS
DNS shared map
CRON
cronfile
Sendmail

FIG. 7B

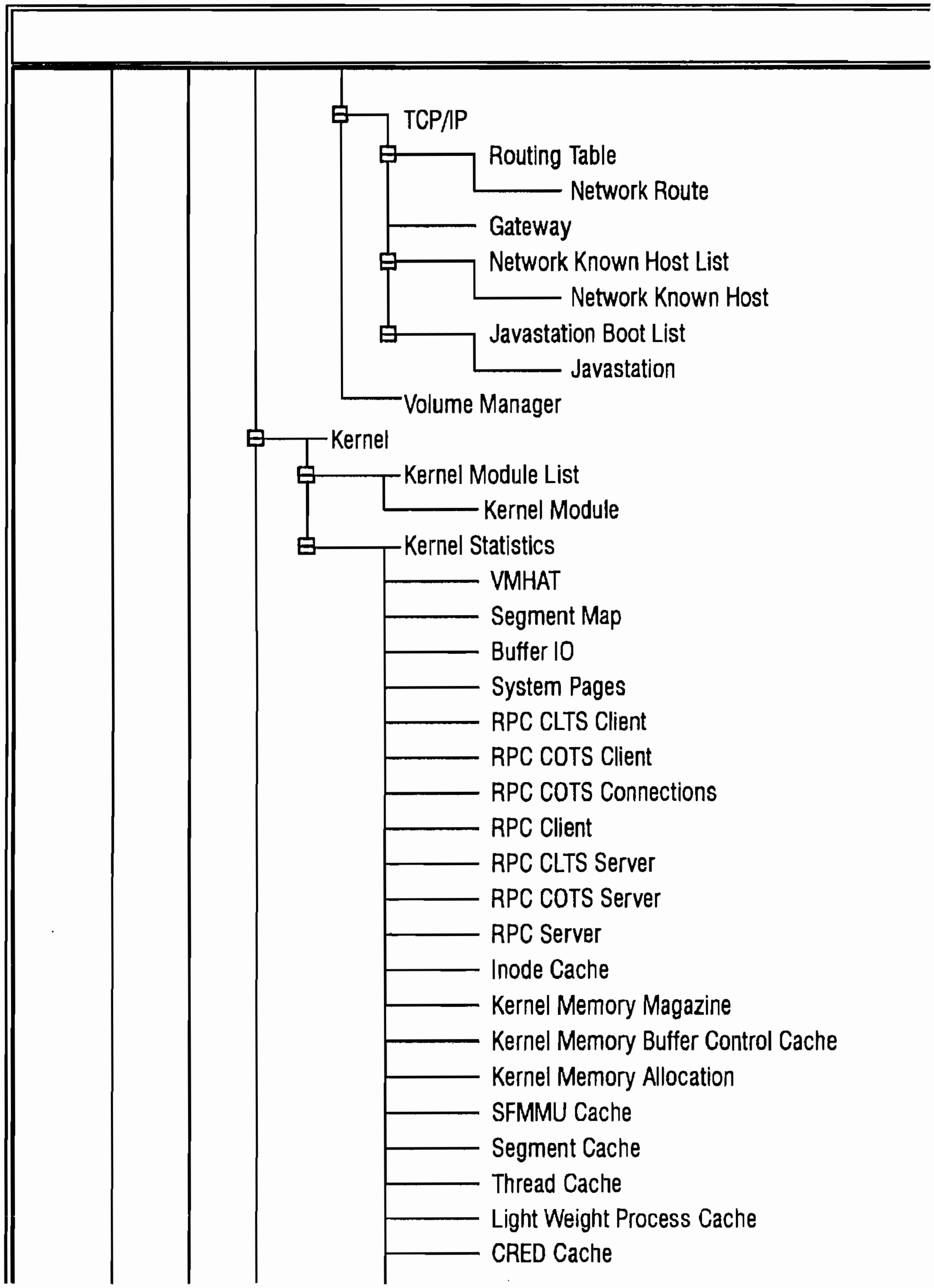

TCP/IP
Routing Table
Network Route
Gateway
Network Known Host List
Network Known Host
Javastation Boot List
Javastation
Volume Manager
Kernel
Kernel Module List
Kernel Module
Kernel Statistics
VMHAT
Segment Map
Buffer IO
System Pages
RPC CLTS Client
RPC COTS Client
RPC COTS Connections
RPC Client
RPC CLTS Server
RPC COTS Server
RPC Server
Inode Cache
Kernel Memory Magazine
Kernel Memory Buffer Control Cache
Kernel Memory Allocation
SFMMU Cache
Segment Cache
Thread Cache
Light Weight Process Cache
CRED Cache

FIG. 7C

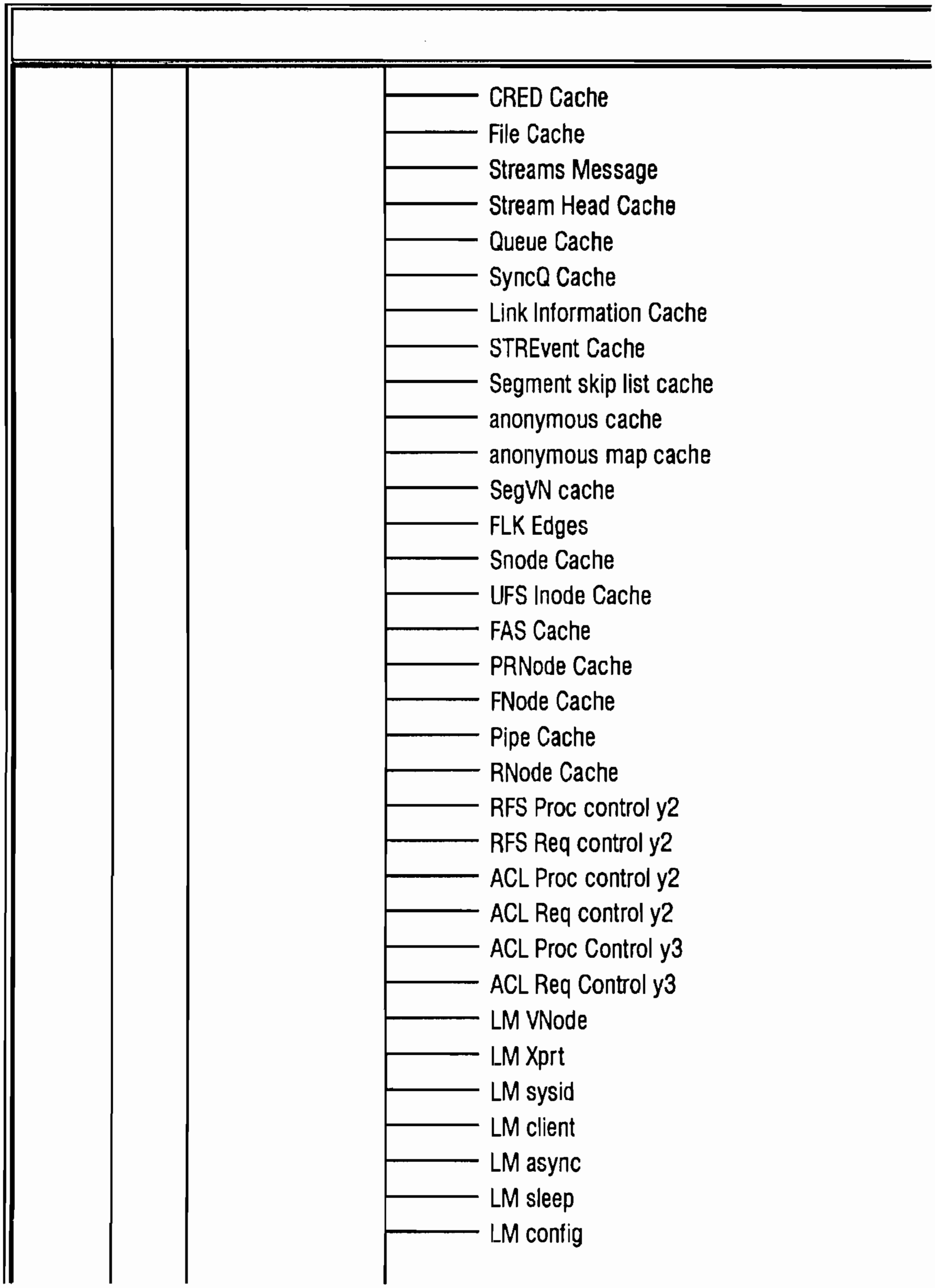

CRED Cache
File Cache
Streams Message
Stream Head Cache
Queue Cache
SyncQ Cache
Link Information Cache
STREvent Cache
Segment skip list cache
anonymous cache
anonymous map cache
SegVN cache
FLK Edges
Snode Cache
UFS Inode Cache
FAS Cache
PRNode Cache
FNode Cache
Pipe Cache
RNode Cache
RFS Proc control y2
RFS Req control y2
ACL Proc control y2
ACL Req control y2
ACL Proc Control y3
ACL Req Control y3
LM VNode
LM Xprt
LM sysid
LM client
LM async
LM sleep
LM config

FIG. 7D

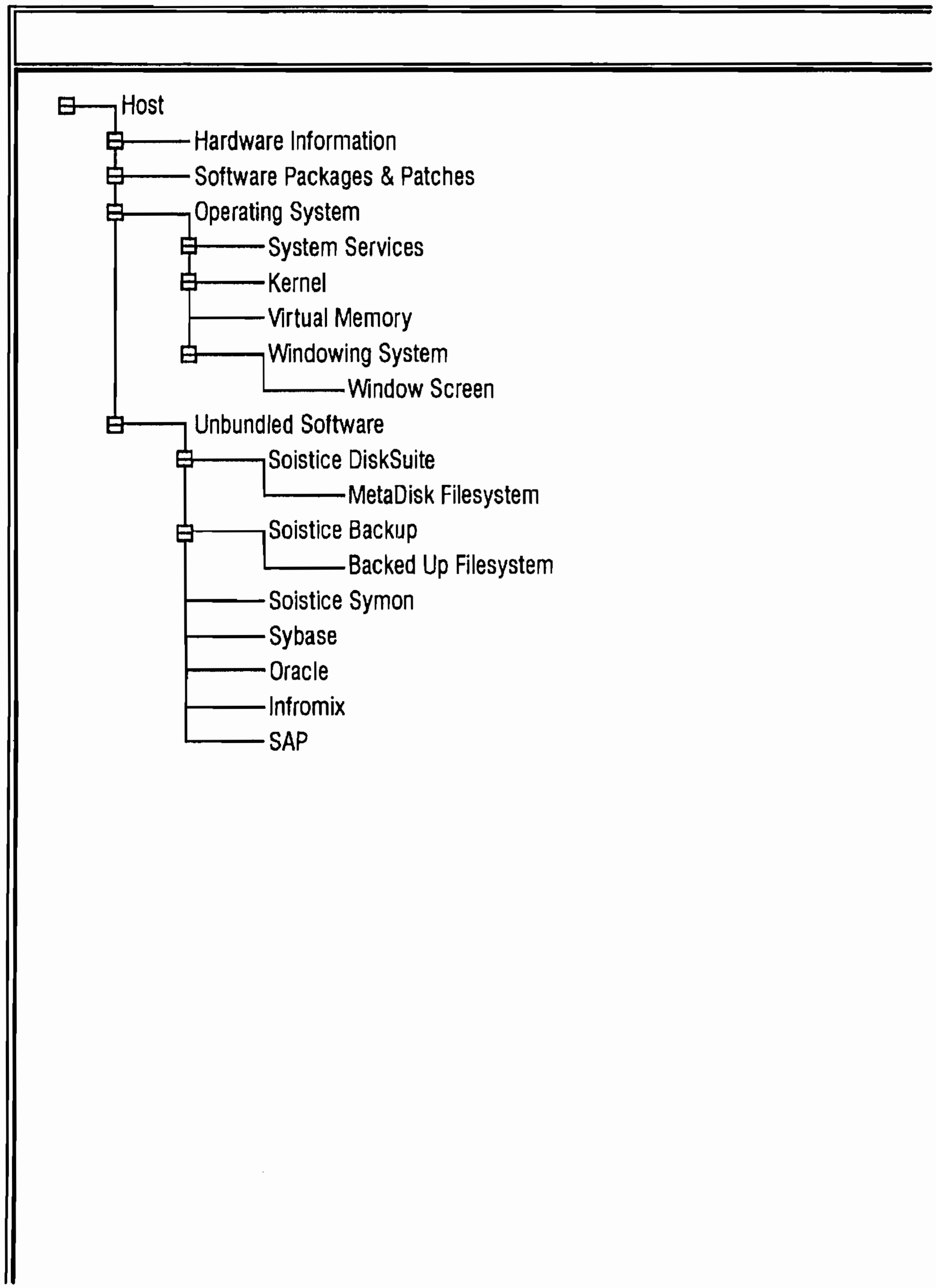

Host
Hardware Information
Software Packages & Patches
Operating System
System Services
Kernel
Virtual Memory
Windowing System
Window Screen
Unbundled Software
Soistice DiskSuite
MetaDisk Filesystem
Soistice Backup
Backed Up Filesystem
Soistice Symon
Sybase
Oracle
Infromix
SAP

FIG. 7E

* Partition	Tag	Flags	Sector	Count	Sector	Mount Directory
c0t0d0s0	2	00	0	2048960	2048959	/
c0t0d0s1	3	01	2048960	262960	2311919	
c0t0d0s2	5	00	0	4154160	4154159	
c0t0d0s7	8	00	2311920	1842240	4154159	/export/home

* Partition	Tag	Flags	Sector	Count	Sector	Mount Directory
c0t1d0s0	2	00	0	62320	62319	/c0t1d0s0.a005WN
c0t1d0s1	3	01	62320	197600	259913	
c0t1d0s2	5	01	0	4154160	4154159	
c0t1d0s0	4	00	259920	3894240	4154159	/c0t1d0s6.a005WN

FIG. 8

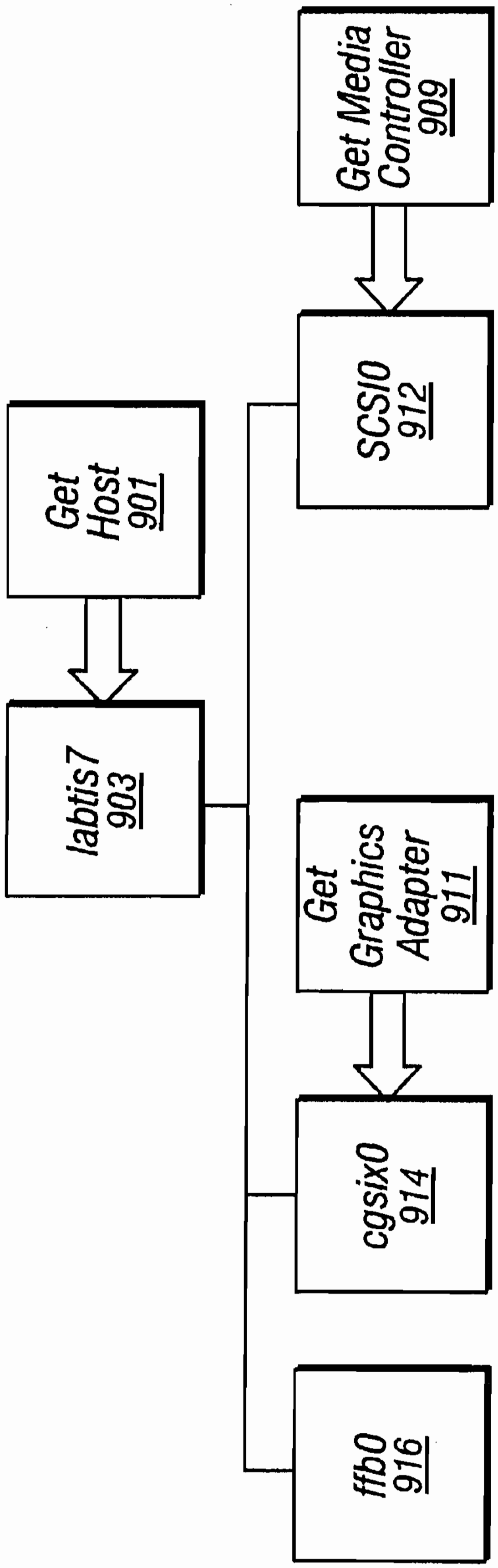

Get Media Controller 909
SCSI0 912
Get Host 901
labtis7 903
Get Graphics Adapter 911
cgsix0 914
ffb0 916

FIG. 9

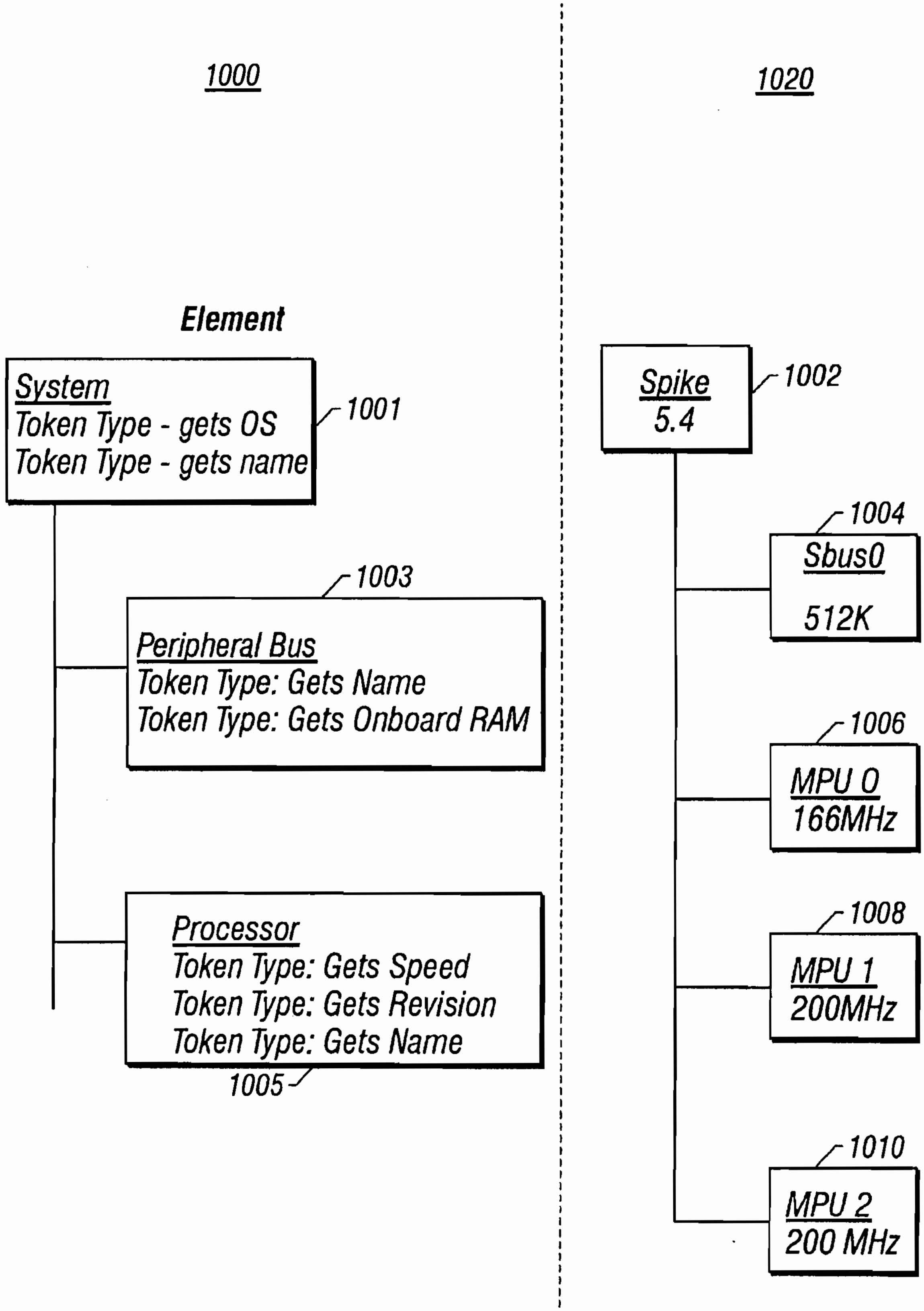

1000
1020
Element
System
Token Type - gets OS
Token Type - gets name
1001
Spike
5.4
1002
1004
Sbus0
512K
1003
Peripheral Bus
Token Type: Gets Name
Token Type: Gets Onboard RAM
1006
MPU 0
166MHz
1008
MPU 1
200MHz
Processor
Token Type: Gets Speed
Token Type: Gets Revision
Token Type: Gets Name
1005
1010
MPU 2
200 MHz

FIG. 10

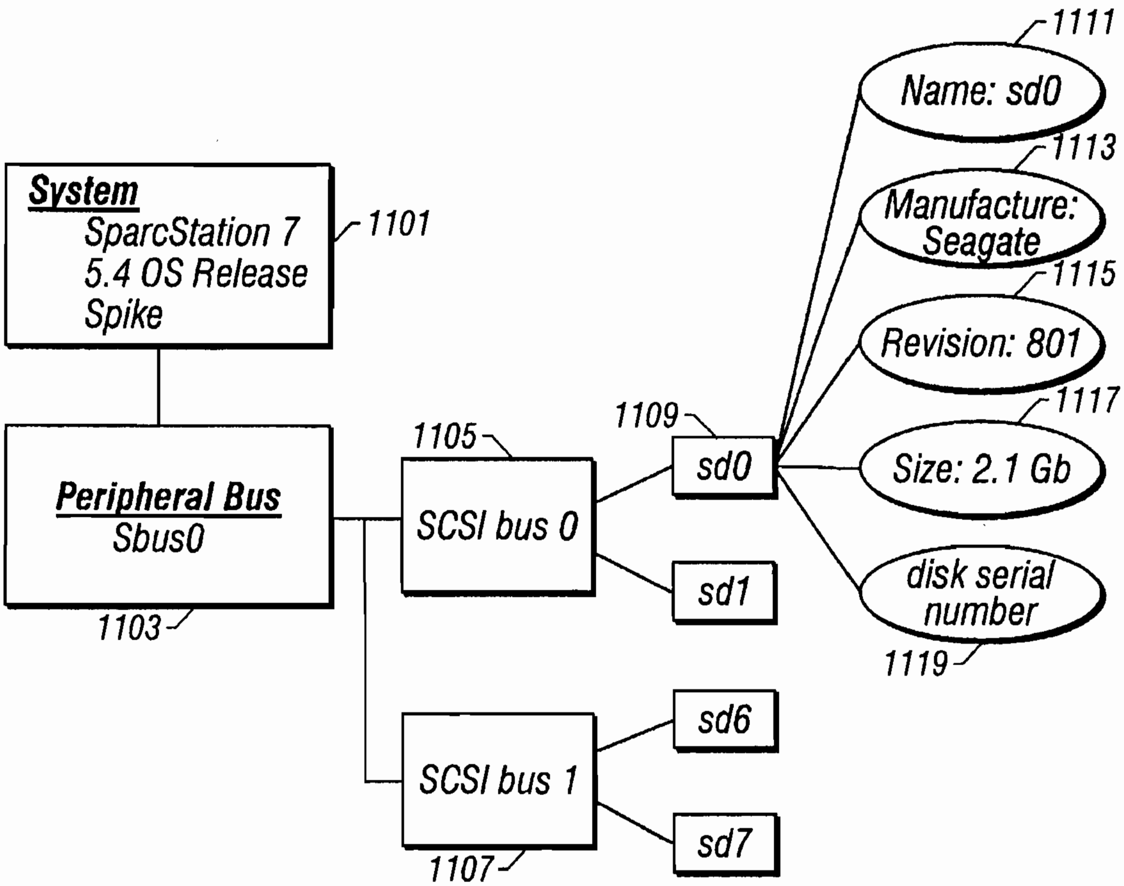

System
SparcStation 7
5.4 OS Release
Spike
1101
Peripheral Bus
Sbus0
1103
1105
SCSI bus 0
1109
sd0
sd1
SCSI bus 1
1107
sd6
sd7
1111
Name: sd0
1113
Manufacture: Seagate
1115
Revision: 801
1117
Size: 2.1 Gb
disk serial number
1119

FIG. 11

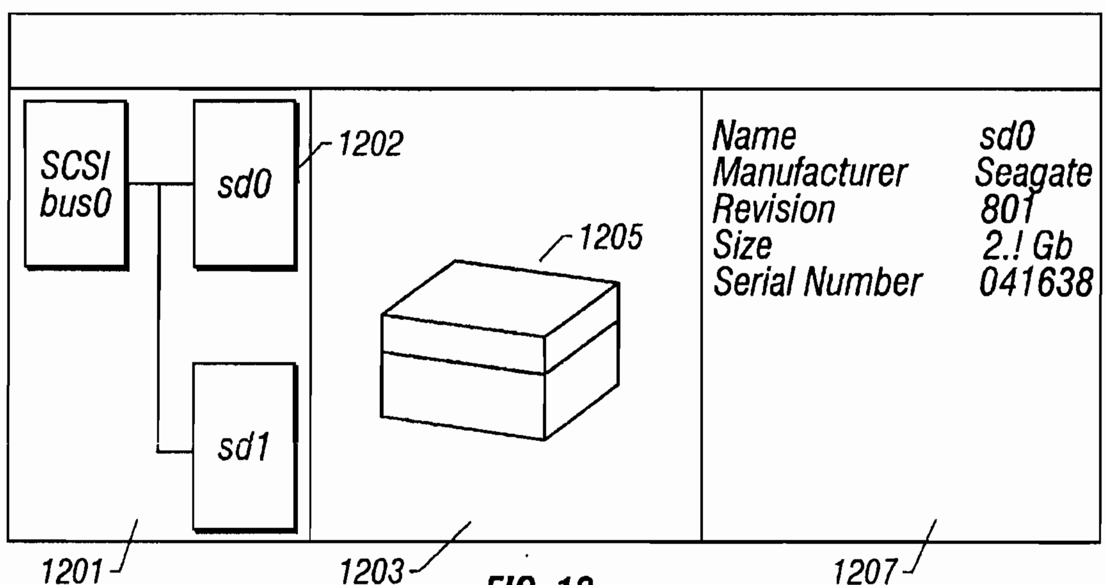

SCSI bus0
sd0
sd1
1202
1205
Name sd0
Manufacturer Seagate
Revision 801
Size 2.! Gb
Serial Number 041638
1201
1203
1207

FIG. 12

REBUILDING COMPUTER STATES REMOTELY

CROSS-REFERENCE TO RELATED APPLICATIONS

This application relates to the following commonly owned co-pending applications, Ser. No. 08/819,501, entitled "AUTOMATIC REMOTE COMPUTER MONITORING SYSTEM", by Michael J. Wookey, filed Mar. 17, 1997, and Ser. No. 08/819,500, entitled "DYNAMIC TEST UPDATE IN A REMOTE COMPUTER MONITORING SYSTEM", by Michael J. Wookey, filed Mar. 17, 1997, which applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to monitoring of computer systems and more particularly to rebuilding the state of a computer system based on diagnostic data from the computer system.

2. Description of the Related Art

Computer systems such as mainframes, minicomputers, workstations and personal computers, experience hardware and software failures that degrade system performance or render the system inoperative. In order to diagnose such failures computer systems include diagnostic capability which provides various types of system diagnostic information.

Computer systems are typically serviced when a failure is noticed either by system diagnostics or by users of the system when the system become partially or completely inoperative. Since computer systems are frequently located at some distance from the support engineers, when problems do occur, a support engineer may access the computer system remotely through a modem in an interactive manner to evaluate the state of the computer system. That remote dial-in approach does allow the support engineer to provide assistance to a remote customer without the delay of traveling to the computer system site. Once connected to the remote computer system, the support engineer can perform such tasks as analyzing hardware and software faults by checking patch status, analyzing message files, checking configurations of add-on hardware, unbundled software, and networking products, uploading patches to the customer system in emergency situations, helping with problematic installation of additional software, running on-line diagnostics to help analyze hardware failures, copying files to or from customer system as needed.

However, there are limitations to such support. For instance, the data size transfer may be limited at the time of failure, due to such factors as modem speed and thus a complete picture of a system may be unavailable. Running diagnostic software during the remote session, if necessary, may adversely impact system performance. Where a system is part of a network, which is commonplace today, the running of diagnostic tests may impact network performance. Where computer systems are being used in a production or other realtime environment, such degradation of system performance is obviously undesirable.

Further, historical data on system performance may not be available in such scenarios. It is therefore impossible to analyze trends or compare system performance, e.g., before and after a new hardware or software change was made to the system. The support engineer is limited to the snapshot of the system based on the diagnostic information available when the support engineer dials in to the system.

It would be advantageous if a support engineer had available complete diagnostic information rather than just a snapshot, However, system diagnostic tests typically generate a significant amount of data and it can be difficult for a support engineer to analyze such data in a raw form. Additionally, service centers typically support a number of different computer systems. Each computer system has its own hardware and software components and thus have unique problems. For example, it is not uncommon for failures to be caused by incorrect or incompatible configuration of the various hardware and/or software components of the particular system. It would be advantageous to provide a remote monitoring diagnostic system that could process, present and manipulate diagnostic data in a structured and organized form and also monitor a number of different computer systems without having prior knowledge of the particular hardware or software configuration of each system being monitored. In order to provide better diagnostic support to computer systems, it would also be advantageous to provide the ability to detect problems in the diagnostic data and to provide proactive monitoring of the diagnostic data in order to better detect and/or predict system problems.

SUMMARY OF THE INVENTION

Accordingly, the present invention provides a method and apparatus to build a representation of the state of a computer, based on diagnostic data, by extracting system information from that diagnostic data and building a component based representation of the computer using the extracted system information. A static tree definition of a computer system is provided which is formed by element types in a fixed hierarchical relationship. A plurality of token types are provided, each of the token types being associated with one of the element types. The token types are component based data types. Respective segments of the incoming data that are defined by respective token types are identified and stored as tokens in a token data base. Each of the tokens has a value field holding a value associated with the element and a parent field referring to an element with which the token is associated. For each element in the static definition, the token data base is searched for associated tokens and a host state is built based on the static state definition and the extracted associated tokens, the elements of the static state definition being realized and given value by their associated tokens.

The method and apparatus of the present invention provides a component based data structure for the diagnostic data that facilitates problem detection as well as proactive monitoring of the monitored computer system. Further, the present invention can build a representation of a monitored computer system without having any prior knowledge of the hardware and software details of the monitored computer system. The present invention also makes the diagnostic data easy to read and accessible to support engineers. Further, the invention can provide support for new computer systems and products in a manner that is more responsive than was previously available.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings, wherein the use of the same reference symbols in different drawings indicates similar or identical items.

FIG. 1*a* shows an exemplary system for rebuilding the state of a computer according to the present invention.

FIG. 1*b* shows an exemplary monitored computer system which runs diagnostic tests on each computer and communicates the results of those tests to the system of FIG. 1*a*.

FIG. 2 details the architecture of a system that rebuilds computer states according to the present invention.

FIG. **3** shows a root and lower branches of a static tree definition of computer system.

FIG. **4** shows additional branches of a static tree definition of a computer system related to components on the CPU-BUS.

FIG. **5** shows additional branches of a static tree definition of a computer system, related to components on the peripheral bus.

FIG. **6** shows additional branches of a static tree definition of a computer system, related to software configuration components.

FIG. **7***a* shows the root and lower branches of a second exemplary tree structure.

FIG. **7***b* shows additional sub elements of the System services element.

FIG. **7***c* shows additional operating system elements.

FIG. **7***d* shows operating system elements related to kernel statistics.

FIG. **7***e* shows unbundled software elements.

FIG. **8** shows an exemplary output of a diagnostic test from which tokens are extracted and used to instantiate the static model exemplified by FIGS. **3**–**6** and FIGS. **7***a*–**7***e*.

FIG. **9** shows an exemplary instantiation of a portion of a static tree.

FIG. **10** shows another example of a tree structure and an instantiation of that tree.

FIG. **11** shows another example of a host state.

FIG. **12** shows how the host state can be displayed to show graphical, and attribute information about the host state.

DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

Referring to FIGS. **1***a* and **1***b*, an exemplary computer system **100**, according to the present invention, receives diagnostic data from a monitored computer system **102**. Monitored computer system **102** runs diagnostic tests, from among tests such as those shown in Table 1 or Table 2, on a periodic basis. The monitored system includes at least one computer and typically includes a plurality of computers **104**–**112** coupled in a network as shown in FIG. **1***b*. The diagnostic tests **116**–**124** are run on the computer system **102** under the control of monitor control software **126**–**134**. The results of those diagnostic tests are automatically provided at periodic intervals to the computer system **100** which monitors computer system **102**. In exemplary computer system **100**, which includes one or more computers and associated storage areas, preferably coupled in a network, incoming diagnostic data from monitored system **102** is received from modem **114** at one of the modems in the modem pool **101**. The incoming data may be received via email or may be a direct modem connection to the monitored system **102** or may be received via other communication channels. The raw diagnostic data is stored in storage **109**. Storage **109** is shown as a single storage unit but may be separate storage units to accommodate the various storage requirements described herein. In order to perform operations on the data received, processor **117** transforms the received incoming data into a structure which can then be analyzed by alert processing computer **119**. Editing capability is provided by a separate computer **121**. Note that the functions may be performed in separate machines or may be combined into one or several computers.

TABLE 1

Class	Test Name	Description
network	automount.files	Automount /etc Files
	automount.nis+	Automount NIS+Files
	automount.nis	Automount NIS Files
	dfshares	NFS shared filesystems
	domainname	Domain name
	etc.defaultdomain	/etc/defaultdomain
	etc.defaultrouter	/etc/defaultrouter
	etc.dfstab	List /etc/dfs/dfstab
	etc.hostnames	/etc/hostname(s)
	etc.hosts	/etc/hosts
	etc.mnttab	List/etc/mnttab
	etc.named.boot	/etc/named.boot
	etc.nsswitch.conf	/etc/nsswitch.conf
	etc.resolv.conf	/etc/resolv.conf
	netstat-an	List all TCP connections
	netstat-in	List network interfaces
	netstat-k	Network interface low-level statistics
	netstat-m	List network routing
	nisdefaults	NIS+ server defaults
	nisstat	NIS+ statistics
	ypwhich	NIS server name
	ypwhich-m	NIS map information
OS	checkcore	Check for core files
	df	Disk Usage
	dmesg	Boot Messages
	framebuffer	Default console/framebuffer
	hostid	Numeric ID of host
	ifconfig	Ethernet/IP configuration
	messages	System messages (/var/adm/messages)
	patches	List system patches
	pkginfo	Software package information
	prtconf	System hardware configuration (Software Nodes)
	prtconf-p	System hardware configuration (PROM Nodes)
	prtdiag	Print diagnostics (Sun-4d systems only)
	sar	System activity reporter
	share	Shared directories
	showrev	Machine and software revision information
	swap	Swap report
	uptime	Local uptime and load average
	whatami	Lengthy system description report
unbundled	fddi-nf_stat	FDDI low-level statistics
	metastat	Online DiskSuite or Solstice DiskSuite
	vxprint	Systems using SPARCstorage Array Volume Manager
	x25_stat	X.25 low-level statistics

TABLE 2

Test Name	Test Name
ps -ef	ypwhich
pkginfo -1	df
vmstat	df -k
showrev -a	mount -v
xdpyinfo	more/etc/dfs/dfstab
netstat -k	cachefsstat
kmemleak (SMCC)	df -1
vtsprobe	df -1k
modinfo	showrev -p
arp -a	nettest -1V (VTS)
netstat -r	dmesg
configd	diskprobe
more/etc/mail/sendmail.cf	disktest -1v (VTS)
crontab -1 (as root)	tapetest -1v (VTS)
more/etc/nsswitch.conf	bpptest -1 v (VTS)
more/etc/resolv.conf	uname -a
niscat -o org_dir	

Referring to FIG. **2**, the architecture of a system according to the present invention, is shown in greater detail. Incoming diagnostic data **201** is received via email or direct modem link (or another communication link) into the monitoring system and stored in raw test data storage area **213**. The test data, which contains information about the software and hardware components in monitored system **102**, is processed by token processing **211** to extract the information associated with hardware and software components in the monitored system. The extracted information is then used to create a representation of the monitored system in host state creator **206** based on the component information. The host state is the state of the monitored system or one computer of the monitored system over the particular time period that the diagnostic tests were run. Further details of the host state will be described further herein.

In order to create a representation of the monitored system, the components contained in the test data are rebuilt into a system hierarchy based on a static hierarchy tree definition. In a preferred embodiment, one static hierarchy tree definition is applicable to all systems which are being monitored. The extracted information about the components in the monitored system are mapped onto the static tree to create the system representation for the monitored system. Thus, the state of the monitored system is rebuilt.

The hierarchy tree is composed of elements. An element can be thought of as a physical or virtual component of a computer system. For example, a computer system may include such components as a disk, a disk partition, a software package, and a patch. An element has tokens associated with it. Thus, a partition element may have a disk percentage token, disk name token, and space available token associated with it. An element definition includes what token types fulfill the element, and give the element value. In one embodiment, an element is an instance of a class of element types as implemented in an object oriented language such as Java.

An exemplary portion of a static tree definition a computer system is shown in FIGS. **3**–**6**. FIG. **3** shows a lower level (closer to the root) elements of the static tree and FIGS. **4**, **5** and **6** show how the tree definition expands. The element host **301** defines the kind of computer that is being monitored. For instance, the host may be a workstation running Solaris or a PC running Windows NT. Attached to host **301** are other physical or virtual components such as CPU bus **303**, monitor **305**, keyboard/mouse **307**, peripheral bus **309** and software configuration **311**. Note that the terms are very general. Each element represents types of components that can be found in a typical computer system.

Referring to FIG. **4**, the computer system further includes additional physical or virtual components on the CPU bus **303**. The additional elements found on the CPU bus include CPU **401**, memory **403** and EEProm **405**. Referring to FIG. **5**, additional components of the static hierarchy tree definition of the computer system can be found under peripheral bus element **309**. Note that the instance of the peripheral bus could be an Sbus. However, the instance could also be a Peripheral Component Interface (PCI) bus. In fact there could be two instances of peripheral bus, SBUS and CPI bus. In some instances there could be more than two peripheral buses. The additional elements found on peripheral bus **309** include display adapter **501**, peripheral adapter **503**, network adapter **505** and port **507**. The peripheral adapter element **503** may be coupled to additional elements such as removable media device element **509**, (e.g., a disk drive, tape or CD drive) or a fixed media device **511**. The fixed media device may be a hard disk drive which can have a further virtual component, partition element **513**. Note the general nature of the static hierarchy system definition. That allows the static definition to be used even for monitored systems that utilize different software and hardware components.

Referring to FIG. **6**, additional software elements under the software configuration element **311** are shown. Included in the software configuration **311** are the operating system element **601**, software services element **603**, patches element **605** and packages element **607**. Additional elements under software services include disk mounts **609**, cron **611**, disk software **613**, naming services **615**, print services **617**, serial port monitors **619** and custom services **621**. The packages element **607** indicate, e.g., what software has been installed on the system. The operating system **601** is further defined by elements **623**–**637**.

The description of the static tree is exemplary. Another tree may be chosen according to the system being monitored. Additionally, the static tree may be modified to reflect hardware and software enhancements to computer systems. The hierarchy tree definition is static in that it does not vary according to the system being monitored. However, the hierarchy tree can be edited in element hierarchy editor **215** to accommodate additions and/or deletions from the hierarchy tree when for instance, a new technology begins to be utilized in the monitored computer systems. One static tree or hierarchy tree definition may be sufficient for most or all monitored systems. However, a hierarchy tree definition could be tailored to the type of computer system that is being monitored to e.g., enhance processing speed. Another exemplary tree structure is shown in FIG. *7a*–*7e*. The tree structure can be seen to include both hardware components and software components.

Thus, given a static definition of a generic computer system such as shown in FIGS. **3**–**6**, or FIGS. *7a*–*7e*, it is possible to build a representation of the actual computer system being monitored utilizing the diagnostic data communicated from the monitored system to the monitoring system.

In order to extract information from the diagnostic data stream, "token types" are utilized. A token type defines each token to have a token name and a test name. A test name comes from the tests shown e.g., in Table 1 or in Table 2, and indicates which test output contains the information for the token. In addition to a token name and a test name, each token has a label and a value. The label for the token gives the token knowledge about what element the token is associated with, i.e., the parent of the token which is an element. The value of the token provides a value extracted from the diagnostic data that gives value to the element.

For instance, assume a disk element exists with a name of "c0t10d0". Assume also that a token exists for such a disk element indicating the number of sectors per cylinder. The name of such a token would be, e.g., "number of sectors per cylinder." The test name in the token would be "vtsprobe" since the output of that test provides the information needed for the number of sectors per cylinder. The label for the token would be "c0t10d0" indicating that token is associated with a particular disk having that name. Finally, the token would have a value which indicates the number of sectors per cylinder. Other tokens could of course be associated with that element. For example, another token associated with that disk element might be a disk manufacturer token that identifies the manufacturer as "Seagate". The value of the token in such an instance would be "Seagate".

Note that one token type can create many tokens from the test data. For example, a "disk name" token type could extract multiple tokens, e.g. the disk names "c0t1d0" and "c0t2d0", from the test data when a particular system has two disks so named.

There are two types of tokens. The first is an element realizing token. Element realizing tokens provide a way to determine whether an element should be included when building a particular host state. For example, a disk name token is an element realizing token. The second type of

token are data tokens which provide additional information about an element that has already been realized, such as the token indicating the number of sector per cylinder. Thus, it can be seen that tokens give value to the elements.

For any particular system, it is preferable to create tokens with as much granularity as possible. Thus, the smallest piece of information that is available about a system from the available diagnostic tests should be included as a token. Representative tokens are included in the description herein. The exact nature of the tokens and the total number of tokens will depend upon the system that is being monitored, including its hardware and operating system, and the diagnostic tests that can be run on the system. Table 3, attached, shows both elements and tokens for an exemplary embodiment of the invention. For each element shown in Table 3, the associated tokens are shown as well as the tests that supply the token information. In addition Table 3 shows the types of computers and operating system releases on which the tests are operable.

An exemplary output of one the diagnostic tests is shown in FIG. **8**. The processing must extract from the output such information as the disk partition ID, last sector, first sector and the like. Examples of the tokens that are extracted for disk partition elements is shown in Table 3 for tokens associated with "SCSI Disk Partition Element". In order to parse through the output of the diagnostic tests a strong textual processing programming language, such as Perl, is utilized.

Note that the preferred implementation of the invention described herein is in an object oriented computer language and more particularly in Java. Nearly all the classes and type definitions described herein extend the type Persistent Object. Persistence is a technique that can be used in object oriented programming to ensure that all memory resident information can be stored to disk at any time. It can be thought of as encoding and decoding. When a persistent object is saved to disk, it is encoded in some manner so that it may be efficiently stored in the appropriate medium. Equally when loading the information back, it is decoded. That allows complex memory structures to be stored easily in databases with minimum disk space impact.

Now that it is understood that a static tree structure is composed of elements which are realized and given value by tokens, the building of a particular representation of a monitored computer system can be more completely described. Referring again to FIG. **2**, the incoming data stream **201** of diagnostic data is stored in raw test data storage area **213**. Token types are stored in storage area **233**. The token types and the diagnostic data are provided to token processing **211**, which is the process of running the token definitions against the incoming data and generating an outgoing stream of tokens which are stored in token data base **207**. In a preferred embodiment the tokens in token data base **207** are stored as a hashtable to provide faster access to subsequent processing steps of building the representation of the system. A hashtable is a common key/element pair storage mechanism. Thus, for the token hashtable, the key to access a location in the hashtable is the token name and the element of the key/element pair would be the token value. Note that because the diagnostic data may include data for multiple computers in a monitored network or subnetwork, one task is to separate the diagnostic data provided to the token processing process **211** according to the computer on which the diagnostic tests were executed. Token types are run against the test output indicated in the test name in the token. For example token types having a test name parameter of "df" are run against "df" test output.

Once all the raw test data has been processed and a completed token data **207** is available, the second set of processing operations to build the representation of the monitored computer may be completed. In order to understand the building of the tree, an examination of several typical features of an element class will provide insight into how an element is used to build a tree.

An element has methods to retrieve the name of the element as well as the various values associated with an element. For example, a disk element includes a method to retrieve a disk ID token which realizes the element as well as having a method to find in the token data base a disk capacity parameter, sectors per track and other tokens such as those shown in Table 3 associated with "SCSI Disk". Those parameters are used to realize a disk element and give it value.

An element of one type is similar to an element of another type. For example; a partition element requires different tokens to provide different values but otherwise is similar to a disk element. The tokens needed to provide value to the partition element may include partition size, partitions used and partition free. Note elements have associated tokens providing a name or ID. As previously described, tokens have both a value and a label. The label or name provides a "tie" for the token. Suppose a disk element is instanced with a name of "c0t1d0". One of its token to be fulfilled is disk size. The token that provides the disk size would have a name of"0t1d0" and a value of 1.2 Gb. The value of 1.2 Gb would be tied to the name "0t1d0".

Referring to FIG. **9**, an example of building a host state based on the elements of the static tree is shown. The term "host state" refers to the representation of the monitored system based on its diagnostic data. The host state essentially describes the state of a system for a given time period. The host state may be viewed as an instantiated element hierarchy based on the raw data that has come in from the remote host. In other words, it is a completed element hierarchy with value. The diagnostic data is collected over a particular time period, so the host state represents the state of the monitored machine over that particular time period, e.g., an hour. The host state is built by starting from the top of the tree element host **301**. The element **301** has methods to retrieve relevant tokens from the token data base **207**. As shown in FIG. **9**, the element **301** is realized as "labtis 7". Because the token data base is a hashtable in the preferred embodiment, the realization of each element is faster. Next element graphics adapter **501** gets graphics adapter cgsix0 **814** and ffb0 **816**. Continuing to build the host state, media controller element gets SCSI0 **812** from the data base. In a preferred embodiment, the host state is built in depth order meaning that each element and all branches of that element are built before another element is built. Thus, referring back to FIG. **5**, for example, everything on peripheral bus **309** would be built before the building of the software configuration **311**. For each element in the static tree, the token data base is searched and the host state is created in element fulfillment processing **205** which requests tokens from token data base **207** in the form of searches for tokens providing realization and value to the static tree.

Once the element fulfillment stage is completed a final token post processing operation takes place in **208**. An element can have a token defined that is the mathematical result of other tokens. For example, a disk space free token is derived from a simple subtraction from a disk used token and a total disk space token. The calculations are completed in this post processing operation **208** to complete the host state.

Note that because the tree definition is static and is intended to be general, not all elements will be found in every host state. Thus, when building the host state, no data will be found in the token data base for a particular element that is lacking in the monitored system. Additionally, in some host states, an element will be found more than once.

Thus, the tree structure provides the flexibility to build host states that look very different.

Once the host state is built, it is saved in host state storage **209**. The storage of the host state provides several advantages. For one, it provides the capability to search back through time and to compare one host state with another host state from a different time or perform trend analysis over time. The host states may be stored for any amount of time for which adequate storage area is available. For example, host states may be stored for a year.

Additionally, the stored host states are used when the diagnostic data is incomplete. There may be occasions when a test has failed to run in the monitored system or has not run before a scheduled communication of data from the monitored system. That may cause problems in the building of the host state from the static tree, especially where the test was one that created elements lower in the tree (i.e. towards the root). Each element can include a value that indicates how critical the element is to the system. If the element is critical, such as a disk, there could be a problem with the system and it should be noticed. If the data is not critical to the system, then older data could be retrieved from the previous host state in time for that particular host. That could be limited by restricting such retrieval to a specified number of times, e.g., 10, or any other number appropriate to the criticality of the element, before marking data as invalid.

Referring again to FIG. **2**, the expert transport **250** provides access to all of the data storage mediums used for the various processes requiring the storage mediums. The communications between processing and storage elements is preferably network based to allow flexibility in implementation as the load of the subsystems may be distributed across machines if need be. Each module can access the expert transport in a very rigid manner making use of the object orientated design facilities provided by Java.

A second example of building a host state is shown in FIG. **10**. Element **1001** has associated token types for the name of the system and the OS. Peripheral bus element **1003** has associated token types which gets the name of the peripheral bus and any onboard RAM. Element **1005**, which is a processor element, has associated token types to provide a name, a revision number and the processor speed. The static definition **1000** creates a host state **1020** where the system is realized as "Spike" with an OS release of 5.4. The peripheral bus is instantiated as Sbus0 with 512 K of RAM. The processor element is instantiated three times as MPU0 **1006**, MPU1 **1008** and MPU2 **1010**. Thus, an example is provided where a single element is realized more than one time in a particular system.

Referring to FIG. **11**, another example of a host state is provided. The system is shown as element **1101** with associated values of being SparcStation2, with a system name Spike and an OS 5.4 release. The system has a peripheral bus, Sbus0, which has two SCSI buses **1105** and **1107**. Attached on SCSI bus 0 are two disks sd0 and sd1. Disk "sd0" has associated tokens, in addition to its name, the manufacturer **1113**, the revision **1115**, the size of the disk, **1117** and the serial number **1119**. As seen in Table 3, for the SCSI disk element, other tokens may be associated with a disk element.

In addition to storing the host state in data base **209**, the system provides a graphical interface to access information about the host state. Referring to FIG. **12**, an exemplary system visualization screen is shown. The tree structure is provided in region **1201** of the screen which graphically represents a portion of the host state shown in FIG. **11**. Tree structures may also be represented in the form shown in FIGS. *7a–7e* or other appropriate form. In addition to displaying the tree structure which provides the user a graphical depiction of the completed element hierarchy for a particular system at a particular time, the screen also provides a graphical image of the particular component which is being viewed. For instance, region **1203** of the screen shows a graphical image **1205** of a disk. Assuming that the viewer had clicked on disk **1202**, sd0, region **1207** shows the attributes or token values associated with the selected element. Thus, the attributes relating to name, manufacturer, revision, size and serial number are all provided. This presents the support engineer with an easily understandable graphical image of the total system, and any particular component of the system that is represented in the host state, along with pertinent attributes.

Referring again to FIG. **2**, the system visualizer **225** receives host states from host states database **209** and customer system information stored in data base **235**. The system visualizer also receives alerts and local configurations relevant to a particular support engineer. The first task that the system visualizer must be to select the particular host that is to be worked upon or viewed. Thus, the system visualizer will have to search the host states database **209**. The visualizer will provide the ability to parse through time to select from all the host states available for a particular system. While each element may have a graphic associated with it, a separate graphic can be used to indicate that a problem exists with a particular element.

In addition to displaying the attributes of an element, which are the values of the tokens associated with the element, the system visualizer provides graphical capability to graph attributes against time. One or more attributes can be selected to be graphed against history. In other words, the same attributes from different instances of the element hierarchy for a particular system can be compared graphically. For example, the amount of disk free over time can be monitored by looking at outputs of the "df" test over a period of time. The df output includes such token values as disk percentage used for a particular partition, partition name and size of partition. The visualizer will extract the tokens representing amount of disk percentage used for a particular set of host states. The host states from which the disk percentage tokens are extracted is determined according to the time period to be viewed. That information can then be visualized by plotting a graph of disk percentage used against time. Also, the visualizer can view different instances of the host state. In other words, the visualizer can view the state of a monitored system at different times. That capability provides a visual interpretation of changes in system configuration. The visualizes accesses the stored multiple instances of the host state of the particular system to provide that capability.

While it is possible for the diagnostic data from the monitored system to come up to the monitoring system in a raw form, it is also possible to do some preprocessing on the data in the monitored system. The preprocessing could translate the diagnostic data to something more easily readable by the monitoring system. As a simple example, the monitored system could eliminate all white space in the test output. The choice of whether to do preprocessing may depend on such considerations as whether the additional load put on the monitored system is a cost that is outweighed by the benefit of simpler processing at the monitoring system.

The description of the invention set forth herein is illustrative, and is not intended to limit the scope of the invention as set forth in the following claims. For instance, while exemplary tests were generally described in terms computers operating in a Unix environment, the invention is also applicable to computer systems utilizing a variety of operating systems. Variations and modifications of the embodiments disclosed herein, may be made based on the description set forth herein, without departing from the scope and spirit of the invention as set forth in the following claims.

TABLE 3

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Host	Hostname		showrev -a	showrev -a			showrev -a	showrev -a	
	Platform	uname -a	uname -a	uname -a		uname -a	uname -a	uname -a	
	Host Id		showrev -a	showrev -a			showrev -a	showrev -a	
	Serial Number	ReMon Extract	ReMon Extract	configd		ReMon Extract	ReMon Extract	configd	
	Data Date	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract
	OS Release		showrev -a	showrev -a				showrev -a	showrev -a
	Kernel Architecture		showrev -a	showrev -a				showrev -a	showrev -a
	Application Architecture		showrev -a	showrev -a			showrev -a	showrev -a	
	Hardware Provider		showrev -a	showrev -a			showrev -a	showrev -a	
	Network Domain		showrev -a	showrev -a			showrev -a	showrev -a	
	Kernel Version		showrev -a	showrev -a			showrev -a	showrev -a	
	Openwindows Version		showrev -a	showrev -a			showrev -a	showrev -a	
	System Clock Frequency		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
Hardware Information	Hardware Information(D)	D	D	D	D	D	D	D	D
	Application Architecture		showrev -a	showrev -a			showrev -a	showrev -a	
System Board	System Board ID		vtsprobe	vtsprobe		D	D	D	
	type	ND	ND	configd		ND	ND	configd	
	State	ND	ND	configd		ND	ND	configd	
	slot number	ND	ND	configd		ND	ND	ND	
	temperature	ND	ND	configd		ND	ND	ND	
	board reference number	ND	ND	configd		ND	ND	configd	
Memory Controller	Memory Controller ID	D	D	configd		D	D	D	
	memory controller model	ND	ND	configd		ND	ND	ND	
Memory	Memory Value		vtsprobe	configd			vtsprobe	configd	
Memory Simm	Memory Simm ID	ND	ND	configd		ND	ND	configd	
	board reference number	ND	ND	configd		ND	ND	configd	
	size	ND	ND	configd		ND	ND	configd	
CPU	CPU Unit ID		vtsprobe	configd			vtsprobe	configd	
	Clock frequency		vtsprobe	configd			vtsprobe	configd	
	CPU model		vtsprobe	configd			vtsprobe	configd	
	CPU dcache size	ND	ND	configd		ND	ND	configd	
	CPU ccache size	ND	ND	configd		ND	ND	configd	
	CPU icache size	ND	ND	configd		ND	ND	configd	
	CPU status	ND	ND	configd		ND	ND	configd	
EEPROM	EEPROM Model	D	D	configd		D	D	D	
	Flash Prom model	ND	ND	configd		ND	ND	ND	
Peripheral Bus	peripheral bus id		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	Model No	ND	ND	configd		ND	ND	configd	
	Registration Number	ND	ND	configd		ND	ND	configd	
Audio Port	Audio Port id		vtsprobe	configd				configd	
Audio Port Error	Audio Port Error String		audio -Iv	audio Iv			audio Iv	audio -iv	
Parallel Port	parallel port id		vtsprobe	configd			vtsprobe	configd	
ParallelPortError	Parallel Port Error String		bpptest -Iv	bpptest -Iv			bpptest -Iv	bpptest -Iv	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Serial Port	serial port id		vtsprobe	configd			vtsprobe	configd	
Diskette	Diskette ID		vtsprobe	configd			vtsprobe	configd	
	Diskette Status	ND	ND	configd		ND	ND	configd	
	Diskette Type	ND	ND	configd		ND	ND	configd	
Peripheral Adaptor	Peripheral Adaptor ID		vtsprobe	configd			vtsprobe	configd	
	peripheral adaptor model name		vtsprobe	configd			vtsprobe	configd	
	peripheral adaptor type		vtsprobe	configd			vtsprobe	configd	
	sbus slot no	ND	ND	configd		ND	ND	configd	
	speed register	ND	ND	configd		ND	ND	configd	
CDROM	CDROM ID		vtsprobe	configd			vtsprobe	configd	
Tape	TAPE ID		vtsprobe	configd			vtsprobe	configd	
	Tape Type		vtsprobe	configd			vtsprobe	configd	
Tape Hardware Errors	Tape HW Error String		tapetest -Iv	tapetest=Iv			tapetest -Iv	tapetest -Iv	
SCSI Disk	SCSI Disk ID		vtsprobe	configd			vtsprobe	configd	
	SCSI Disk Sectors per track		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI disk firmware rev		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI disk serial number		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Sectors per Cylinder		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Sun ID		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Cylinders		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Capacity		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Software Controller		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Accessible Cylinders		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Tracks per Cylinder		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Bytes per Sector		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Vendor		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
SCSI Disk Error	Disk Error String		disktest-Iv (o)	disktest -Iv (o)			disktest -Iv (o)	disktest -Iv (o)	
SCSI Disk Partition	SCSI DIsk Partition ID	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
	SCSI DIsk Partition last sector	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
	SCSI Disk Partition Sector Count	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
	Scsi Disk Partition First Sector	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
SCSI Disk Bad Block	SCSI BAD Block ID	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg
	Time occurred	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg
Network Adaptor	Network Adapator ID		vtsprobe	configd			vtsprobe	configd	
	Internet Address		vtsprobe	configd			vtsprobe	configd	
	sbus slot no.	ND	ND	configd		ND	ND	configd	
Network Hardware Error	Network HW Error String		nettest -Iv (o)	nettest -Iv (o)			nettest -Iv (o)	nettest -Iv (o)	
Serial OpticalChannel Processor Host Adaptor	Serial Optical Processor ID	ND	ND	configd		ND	ND	configd.	
	Sbus slot number	ND	ND	configd		ND	ND	configd	
	SO model No.	ND	ND	configd		ND	ND	configd	
Storage Array									
Storage Array Disk									

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Storage Array Partition									
Graphics Adaptor	Graphics Adaptor ID		vtsprobe	configd			vtsprobe	configd	
	graphics adaptor model no	ND	ND	configd		ND	ND	configd	
	sbus slot number	ND	ND	configd		ND	ND	configd	
Monitor	Monitor Dtype(D)	D	D	D	D	D	D	D	D
Serial Port Expander									
System Board PSU	Power Supply ID	ND	ND	configd		ND	ND	ND	
	Power Supply Wattage	ND	ND	configd		ND	ND	ND	
	Power Supply Status	ND	ND	configd		NP	ND	ND	
System Power Supply	System Power Supply (D)	D	D	D		D	D	D	
Software package & patches	Software Packages & packages(D)	D	D	D	D	D	D	D	D
Software Package	Software Package ID		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Vendor		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Files		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Category		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo-1	
	Package Architecture		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Status		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Base Directory		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Version		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package pstamp		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Installation date		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
	Package Name		pkginfo -1	pkginfo -1			pkginfo -1	pkginfo -1	
Software Patch	Software Patch ID	showrev -p	showrev -p	showrev -p		showrev -p	showrev -p	showrev -p	
	Revision ID	showrev -p	showrev -p	showrev -p		showrev -p	showrev -p	showrev -p	
Operating System	Operating System ID	showrev -a	showrev -a	showrev -a		showrev -a	showrev -a	showrev -a	
	OS Release Type	showrev -a	showrev -a	showrev -a		showrev -a	showrev -a	showrev -a	
	OS Release Version	showrev -a	showrev -a	showrev -a		showrev -a	showrev -a	showrev -a	
System Services	System Services(D)	D	D	D	D	D	D	D	D
Unix Filesystems	Unix Filesystems(D)	D	D	D	D	D	D	D	D
Local Filesystems (Boot)	Local Filesystem name	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	Associative Device Name	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	mount point	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	FSCK pass	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	mount options	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
Local Filesystem (Current)	Local Current Filesystem Name	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Associative Device Name	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	mount point	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	mount options	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Date mounted since	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Total Kbytes available	df -lk	df -lk	df -lk	df-lk	df-lk	df-lk	df-lk	df-lk

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Total Kbytes used	df -Ik	df -Ik	df -Ik	df-Ik	df-Ik	df-Ik	df-Ik	df-Ik
	Percentage Capacity	df -Ik	df -Ik	df -Ik	df-Ik	df-Ik	df-Ik	df-Ik	df-Ik
	Files used	df -I	df -I	df -I	df-I	df-I	df-I	df-I	df-I
Cache FS	Cachefs Daemon	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Cached Filesystem (boot)	Cached Filesystem id	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	Filesystem cached	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	mount option	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
Cached File System(Current)	Cached Filesystem ID	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Filesystem cached	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	cache bits	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat
	cache misses	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat
	cache modifies	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat
NFS Server	NFS Server exists	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Exported Filesystem (boot)	Exported Filesystem name	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab
	Export options	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab
	Export Description	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab	more /etc/ dfs/dfstab
Exported Filesystem (current)	Exported filesystem name	share	share	share	share	share	share	share	share
	Exported file systemm options	share	share	share	share	share	share	share	share
	Exported FS description	share	share	share	share	share	share	share	share
NFS Client	Filesystems being mounted	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Mounted Filesystem (Boot)	Filesystem Name	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	remote machine	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	remote filesystem	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	Mount type (kerberos/NFS)	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	mount point	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
	mount permissions	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab	more /etc/ vfstab
Mounted Filesystem (Current)	Filesystem Name	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Remote machine	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	remote filesystem	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	mount type	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	mount point	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Date mounted since	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v	mount -v
	Total Kbytes available	df -k	df -k	df -k	df -k	df -k	df -k	df -k	df -k
	Total Kbytes used	df -k	df -k	df -k	df -k	df -k	df -k	df -k	df -k

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Percentage Capacity	df -k	df -k	df -k	df -k	df -k	df -k	df -k	df -k
	Files used	df	df	df	df	df	df	df	df
NIS Server	NIS Server in existance	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	domainname	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a
NIS client	NIS client software exists	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	domainname	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a
	Server Bound to	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich
NIS+ Master	configured as NIS+ Master	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	domainname	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	access rights	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
NIS+ Replica	configured as a NIS+ Replica	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	domainname	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	access rights	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	NIS+ Master	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
NIS+ Client	configured as aNIS+ client	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	domainname	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
	NIS+ Master	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
NIS+ Search Path	NTS+ Search Path	niscat -o org_dir	niscat -o org_dir	niscat -o org_dir		niscat -o org_dir	niscat -o org_dir	niscat -o org_dir	
DNS client	DNS being used	ps -ef	ps -ef	ps -ef		ps -ef	ps -ef	ps -ef	
DNS resolve hosts	DNS resolve host id	more /etc/ resolv.conf	more /etc/ resolv.conf	more /etc/ resolv.conf		more /etc/ resolv.conf	more /etc/ resolv.conf	more /etc/ resolv.conf	
NameService Configuration	String Dummy Token	D	D	D	D	D	D	D	D
Password Map Resolve Type	Name Service map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
Group MapResolve Type	Name Service map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
Hosts Map Resolve Type	Name Service map resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
Protocols Map Resolve Type	Name Service map resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
RPC Map Resolve Type	Name Service map resolved by	more /etc/	more /etc/	more /etc/		more /etc/	more /etc/	more /etc/	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
		nss- witch.conf	nss- witch.conf	nss- witch.conf		nss- witch.conf	nss- witch.conf	nss- witch.conf	
Ethers Map Resolve Type	Name Service map resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
Netmasks Map Resolve Type	Name Service map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
boorparams Map Resolve Type	Name Service map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
publickey Map Resolve Type	Name Service map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
netgroup Map Resolve Type	Name Service Map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
automount Map resolve Type	Name Service Map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
aliases Map resolve Type	Name Service Map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
services Map resolve Type	Name Service Map Resolved by	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
sendmailvars resolve Type	Name Service Map Resolved By	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf		more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	more /etc/ nss- witch.conf	
CRON	cron is running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
root cronjob	cronjob name	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1
	cronjob time control string	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1
	cronjob execution string	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1	crontab -1
Sendmail	sendmail is running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	marjor relay mailer	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf
	major relay host (DR)	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf
	major relay host (CR)	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf	more /etc/ mail/send- mai.cf
TCP/IP	Name of Ethernet board		vtsprobe	configd			vtsprobe	configd	
	Internet Address	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a
Routing Table	Dummy Token	D	D	D	D	D	D	D	D

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Network Route	Network Route Destination	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r
	Gateway	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r
	Flags	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r
	use	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r	netstat -r
Network Known host list	Dummy Token	D	D	D	D	D	D	D	D
Network Known host	Known IP address	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a
	Network Mask	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a
	Physical Address	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a	arp -a
Javastation boot list									
Javastation									
Volume manager	Create if running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Kernel	Kernel Architecture	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a	showrev -a
	Kernel memory		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
Kernel Memory Leak	Value leaked		kmemleak	kmemleak			kmemleak	kmemleak	
Kernel module list	Dummy Token	D	D	D	D	D	D	D	D
kernel module	Kernel Module Name	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
	Kernel Module Load address	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
	Kernel Module Info	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
	Kernel Module Revision	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
kernel statistics	dummy Token	D	D	D	D	D	D	D	D
VMHAT	dummy token	D	D	D	D	D	D	D	D
	VH CTX Free		netstat -k	netstat -k			netstat -k	netstat -k	
	VH CTX Dirty		netstat -k	netstat -k			netstat -k	netstat -k	
	VH CTX steal		netstat -k	netstat -k			netstat -k	netstat -k	
	VH TTE load		netstat -k	netstat -k			netstat -k	netstat -k	
	VH page faults		netstat -k	netstat -k			netstat -k	netstat -k	
	VH steal count		netstat -k	netstat -k			netstat -k	netstat -k	
Segment map	Dummy test	D	D	D	D	D	D	D	D
	Faults		netstat -k	netstat -k			netstat -k	netstat -k	
	Faulta		netstat -k	netstat -k			netstat -k	netstat -k	
	getmap		netstat -k	netstat -k			netstat -k	netstat -k	
	Page Create		netstat -k	netstat -k			netstat -k	netstat -k	
Buffer IO	Dummy Token	D	D	D	D	D	D	D	D
	Buffer Cache Lookups		netstat -k	netstat -k			netstat -k	netstat -k	
	Buffer Cache bits		netstat -k	netstat -k			netstat -k	netstat -k	
	Waits for Buffer Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
	Duplicate Buffers found		netstat -k	netstat -k			netstat -k	netstat -k	
System pages	Dummy Token	D	D	D	D	D	D	D	D
	Physical Memory		netstat -k	netstat -k			netstat -k	netstat -k	
	Inalloc		netstat -k	netstat -k			netstat -k	netstat -k	
	nFree		netstat -k	netstat -k			netstat -k	netstat -k	
	kernel base		netstat -k	netstat -k			netstat -k	netstat -k	
	econtig		netstat -k	netstat -k			netstat -k	netstat -k	
	free memory		netstat -k	netstat -k			netstat -k	netstat -k	
	available rmem		netstat -k	netstat -k			netstat -k	netstat -k	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	pages free		netstat -k	netstat -k			netstat -k	netstat -k	
	pages locked		netstat -k	netstat -k			netstat -k	netstat -k	
RPC CLTS Client	Dummy Token	D	D	D	D	D	D	D	D
	Calls		netstat -k	netstat -k			netstat -k	netstat -k	
	Badcalls		netstat -k	netstat -k			netstat -k	netstat -k	
	badxids		netstat -k	netstat -k			netstat -k	netstat -k	
	timeouts		netstat -k	netstat -k			netstat -k	netstat -k	
RPC COTS Client	Dummy Token	D	D	D	D	D	D	D	D
	Calls		netstat -k	netstat -k			netstat -k	netstat -k	
	badcalls		netstat -k	netstat -k			netstat -k	netstat -k	
	badxids		netstat -k	netstat -k			netstat -k	netstat -k	
	interrupts		netstat -k	netstat -k			netstat -k	netstat -k	
RPC COTS Connections	Dummy Token	D	D	D	D	D	D	D	D
	Write Queue		netstat -k	netstat -k			netstat -k	netstat -k	
	Server		netstat -k	netstat -k			netstat -k	netstat -k	
	status		netstat -k	netstat -k			netstat -k	netstat -k	
RPC Client	Dummy Token	D	D	D	D	D	D	D	D
	calls		netstat -k	netstat -k			netstat -k	netstat -k	
	badcalls		netstat -k	netstat -k			netstat -k	netstat -k	
	re transmits		netstat -k	netstat -k			netstat -k	netstat -k	
	badxids		netstat -k	netstat -k			netstat -k	netstat -k	
	can't send		netstat -k	netstat -k			netstat -k	netstat -k	
RPC CLTS Server	Dummy Token	D	D	D	D	D	D	D	D
	Calls		netstat -k	netstat -k			netstat -k	netstat -k	
	badcalls		netstat -k	netstat -k			netstat -k	netstat -k	
	xdr call		netstat -k	netstat -k			netstat -k	netstat -k	
RPC COTS Server	Dummy Token	D	D	D	D	D	D	D	D
	calls		netstat -k	netstat -k			netstat -k	netstat -k	
	badcalls		netstat -k	netstat -k			netstat -k	netstat -k	
	xdr call		netstat -k	netstat -k			netstat -k	netstat -k	
RPC Server	Dummy Token	D	D	D	D	D	D	D	D
	calls		netstat -k	netstat -k			netstat -k	netstat -k	
	bad calls		netstat -k	netstat -k			netstat -k	netstat -k	
	xdr calls		netstat -k	netstat -k			netstat -k	netstat -k	
Inode cache	Dummy Token	D	D	D	D	D	D	D	D
	size		netstat -k	netstat -k			netstat -k	netstat -k	
	maxsize		netstat -k	netstat -k			netstat -k	netstat -k	
	hits		netstat -k	netstat -k			netstat -k	netstat -k	
	misses		netstat -k	netstat -k			netstat -k	netstat -k	
	mallocs		netstat -k	netstat -k			netstat -k	netstat -k	
Kernel Mmory magazine	Magazine ID		netstat -k	netstat -k			netstat -k	netstat -k	
	Buffer Size		netstat -k	netstat -k			netstat -k	netstat -k	
	buffer available		netstat -k	netstat -k			netstat -k	netstat -k	
	alloc fail		netstat -k	netstat -k			netstat -k	netstat -k	
kernel memory buffer control cache	Dummy Token	D	D	D	D	D	D	D	D

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Buffer Size		netstat -k	netstat -k			netstat -k	netstat -k	
	buffer available		netstat -k	netstat -k			netstat -k	netstat -k	
kernel memory allocation	Kernel memory allocation ID		netstat -k	netstat -k			netstat -k	netstat -k	
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	buffer size		netstat -k	netstat -k			netstat -k	netstat -k	
	buffer total		netstat -k	netstat -k			netstat -k	netstat -k	
SFMMU Cache	SFMMU Cache ID		netstat -k	netstat -k			netstat -k	netstat -k	
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Failed Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
Segment Cache	Dummy Token		netstat -k	netstat -k			netstat -k	netstat -k	
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Failed Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
Thread Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat -k	netstat -k			netstat -k	netstat -k	
	Failed Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
Leight Weight Process Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat -k	netstat -k			netstat -k	netstat -k	
	Failed Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
CRED Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat -k	netstat -k			netstat -k	netstat -k	
	Failed Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
Thread Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat -k	netstat -k			netstat -k	netstat -k	
	Failed Allocations		netstat -k	netstat -k			netstat -k	netstat -k	
	global allocations		netstat -k	netstat -k			netstat -k	netstat -k	
Streams Message	Stream message ID		netstat -k	netstat -k			netstat -k	netstat -k	
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
Stream Head Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
Queue Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
Sync Q Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
Streams Message	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
anonymous Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
anonymous map Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
seg VN Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
UFS InodeCache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
FASCache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
PipeCache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
LM sysid	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
LM client	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat -k	netstat -k			netstat -k	netstat -k	
	Allocation Failures		netstat -k	netstat -k			netstat -k	netstat -k	
Virtual Memory	Total Virtual Memory size		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	Virtual Memory Free	vmstat	vmstat	vmstat		vmstat	vmstat	vmstat	
Page Fault	Page fault id	vmstat	vmstat	vmstat		vmstat	vmstat	vmstat	
Windowing System	Windowing system revision	showrev -a	showrev -a	showrev -a		showrev -a	showrev -a	showrev -a	
	Display Size	xdpyinfo	xdpyinfo	xdpyinfo		xdpyinfo	xdpyinfo	xdpyinfo	
	Depth of root window	xdpyinfo	xdpyinfo	xdpyinfo		xdpyinfo	xdpyinfo	xdpyinfo	
	resolution	xdpyinfo	xdpyinfo	xdpyinfo		xdpyinfo	xdpyinfo	xdpyinfo	
Process Table	Dummy Token	D	D	D	D	D	D	D	D
Process	process name	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	time	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	process id	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	parent id	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Unbundled Software	Dummy Token	D	D	D	D	D	D	D	D
Solstice DiskSuite									
MetaDisk Partition									
Solstice Backup									
Solstice Backup partition									
Solstice Symon	installed	ND	ND	pkginfo -1	pkginfo -1	ND	ND	pkginfo -1	pkginfo -1
	Symond running	ND	ND	ps -ef	ps -ef	ND	ND	ps -ef	ps -ef
	Kernel reader running	ND	ND	ps -ef	ps -ef	ND	ND	ps -ef	ps -ef
	Log scanner running	ND	ND	ps -ef	ps -ef	ND	ND	ps -ef	ps -ef

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Sybase	Sybase datasrver running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
	sybase backup dataserver running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Oracle	Oracle server running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
Informix	Informix server running	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef	ps -ef
SAP									

Key
(D) - Dummy Token
(M) - Output of test must be modified as it is different across OS releases
(ND) - No data available
*Blank spaces means untested or unknown tests

What is claimed is:

1. A method of rebuilding the state of a computer system, comprising:

providing a static tree structure representing a general computer system;

extracting component information indicating hardware and software components of the computer system, from diagnostic data of the computer system; and

generating a representation of the computer system according to the tree structure and the component information, wherein the computer system is part of a first computer system and the diagnostic data is communicated from the first computer system to a second computer system, the second computer system rebuilding the state of the computer system.

2. The method as recited in claim 1 wherein the static tree structure includes element types in a fixed hierarchical relationship, the element types representing the hardware and software components of the computer system.

3. The method as recited in claim 2 further comprising extracting the component information as a plurality of tokens, the tokens being respective instances of respective token types, each of the token types having a value of one aspect of the component information and an indication of an association with one of the elements in the static tree.

4. The method as recited in claim 3 further comprising building a token data base of the component information from plurality of tokens.

5. The method as recited in claim 4 wherein the token data base is created as a hashtable.

6. The method as recited in claim 4 wherein the generating of the representation of the computer system further comprises:

for each respective element type in the static tree structure, searching the token data base for at least one token that refers to the respective element type;

building the representation of the computer system having instantiated elements according to the tokens in the token data base.

7. The method as recited in claim 6 wherein a respective element type is instantiated when a realizing token is found in the diagnostic data associated with the respective element type.

8. The method as recited in claim 7 wherein the realizing token provides a name for the element type.

9. The method as recited in claim 6 wherein a system representation is built by identifying all tokens in the token data base associated with one element before searching the token data base for tokens referring to another element type.

10. The method as recited in claim 6 wherein at least one of the elements has a token defined to be a result of a mathematical operation on other tokens.

11. The method as recited in claim 6 wherein the representation of the computer system has less than all of all the element types in the static tree structure.

12. The method as recited in claim 6 wherein the representation of the computer system has instantiated more than one of at least one of the element types in the static tree structure.

13. The method as recited in claim 3 wherein multiple tokens are generated from a single token type.

14. The method as recited in claim 1 wherein the diagnostic data is communicated a plurality of times and a system representation is built for each communication of diagnostic test results, each system representation being a host state.

15. The method as recited in claim 14 wherein when diagnostic data is missing from a communication of diagnostic data, missing data is filled in based on a previous host state saved in a data base.

16. A data structure for representing a host state of a monitored computer system, the data structure being encoded in computer readable media and comprising:

a plurality of elements in a fixed hierarchical structure representing components of the monitored computer system; and

at least one token respectively associated with each of the elements, the at least one token providing a first expression representing a value of an associated element and a second expression representing the name of the associated element.

17. The data structure as recited in claim 16 wherein the hierarchical structure includes hardware and software elements.

18. The data structure as recited in claim 17 wherein each of the hardware and software elements represents one of a physical and virtual component of the monitored computer system.

19. A computer program stored on computer readable media and operable in a monitoring computer system to:

provide a static tree structure representing a general computer system;

extract component information indicating hardware and software components of a monitored computer system, from diagnostic data of the monitored computer system; and

generate a representation of the monitored computer system according to the tree structure and the component information, wherein the diagnostic data is communicated from the monitored computer system to the monitoring computer system for rebuilding the state of the general computer system.

20. The computer program as recited in claim 19 wherein the static tree structure includes element types in a fixed hierarchical relationship, the element types representing hardware and software components of the monitored computer system.

21. The computer program as recited in claim 20, wherein the component information is extracted as a plurality of tokens, the tokens being respective instances of respective token types, each of the token types having a value of one aspect of the component information and an indication of an association with one of the elements in the static tree.

22. The computer program as recited in claim 21, the computer program being further operable to build a token data base of the component information from plurality of tokens.

23. The computer program as recited in claim 22, wherein, the computer program is operable to search the token data base for at least one token that refers to the respective element type, for each respective element type in the static tree structure; and

is operable to generate the representation of the monitored computer system to have instantiated elements according to the tokens in the token data base.

24. The computer program method as recited in claim 23 wherein a respective element type is instantiated when a realizing token is found in the diagnostic data associated with the respective element type.

25. An apparatus for rebuilding the state of a monitored computer system, comprising:

a first data storage storing a static tree structure representing a general computer system;

a second data storage storing component information indicating hardware and software components of the monitored computer system;

a monitoring computer system, coupled to the first and second data storage, operable to generate a host state of

the monitored computer system according to the static tree structure and the component information, and

a third data storage coupled to the monitoring computer system, storing each generated host state of the monitored computer system for a predetermined period of time, the third data storage thereby holding a history of host states for the monitored computer system.

26. The apparatus as recited in claim **25** wherein the static tree structure includes element types in a fixed hierarchical relationship, the element types representing the hardware and software components of the monitored computer system.

27. The apparatus as recited in claim **26** wherein the second data storage stores the component information as a plurality of tokens, each token including a value representing one aspect of the component information and an indication of an association with one of the elements in the static tree.

28. The method as recited in claim **27** wherein the plurality of tokens are stored in the form of a hashtable.

29. The apparatus as recited in claim **27** further comprising:

a fourth data storage receiving and storing diagnostic data of the monitored computer system;

a fifth data storage storing token types defining tokens to be found in the diagnostic data, each token being an instantiation of one of the token types; and wherein

the monitoring computer system is coupled to the fourth and fifth data storage areas and operable compare the token types to the diagnostic data to extract component information from the diagnostic data.

* * * * *

EXHIBIT C

(12) **United States Patent**
Wookey et al.

(10) **Patent No.: US 6,182,249 B1**
(45) **Date of Patent: *Jan. 30, 2001**

(54) **REMOTE ALERT MONITORING AND TREND ANALYSIS**

(75) Inventors: **Michael J. Wookey**, Sunnyvale; **Kevin L. Chu**, Palo Alto, both of CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Palo Alto, CA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **08/854,788**

(22) Filed: **May 12, 1997**

(51) **Int. Cl.**[7] **G06F 11/30**

(52) **U.S. Cl.** **714/47**; 714/57; 714/37; 714/40

(58) **Field of Search** 714/47, 57, 37, 714/38, 39, 40, 49, 46

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,567,560	1/1986	Polis et al.	364/184
4,637,013	1/1987	Nakamura	370/85
4,709,365	11/1987	Beale et al.	371/11
5,101,402	3/1992	Chiu et al.	370/17
5,155,847	10/1992	Kirouac et al.	395/600
5,299,312	3/1994	Rocco, Jr.	395/200
5,307,354	4/1994	Cramer et al.	371/11.2
5,400,246	3/1995	Wilson et al.	364/146
5,471,399	11/1995	Tanaka et al.	364/491
5,487,169	1/1996	Vraney et al.	395/700
5,491,791	2/1996	Glowny et al.	395/183.13
5,495,610	2/1996	Shing et al.	395/600
5,539,869	7/1996	Spoto et al.	395/154
5,600,796	2/1997	Okamura et al.	395/200.11
5,655,081	8/1997	Bonnell et al.	395/200.32
5,668,944 *	9/1997	Berry	395/184.01
5,696,486	12/1997	Poliquin et al.	340/506

(List continued on next page.)

OTHER PUBLICATIONS

Sun Microsystems, "Sun VTS 2.1 User's Guide", USA, Aug. 1997, Revision A.

"Remote Systems Diagnostics Installation & User Guide, Remote Systems Monitoring (SunReMon™), Remote Dial-in Analysis (SunRDA™)," Release 1.0.1, Sun Microsystems, Mountain View, California, Nov. 1996, (116 pages).

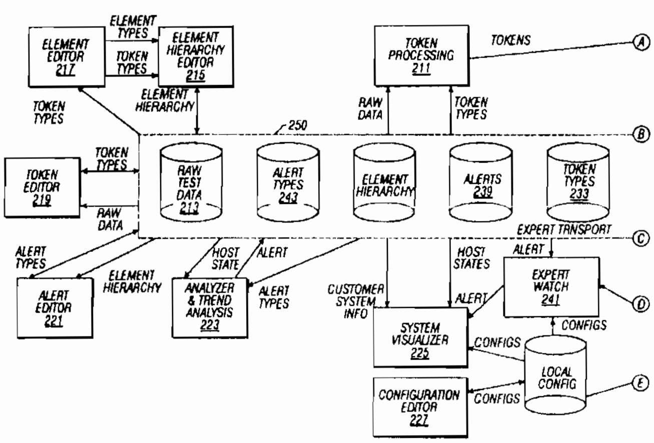
"Solstice™ SyMON™ User's Guide," Revision A, Sun Microsystems Computer Company, Mountain View, California, May 1996 (116 Pages).

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Scott T. Baderman
(74) *Attorney, Agent, or Firm*—Skjerven Morrill MacPherson LLP

(57) **ABSTRACT**

A monitoring system generates alerts indicating predefined conditions exist in a computer system. Alerts are generated by comparing alert definitions to a host state representing the state of the hardware and software components of a computer system. to determine if conditions defined in the alert definitions exist in the host state; and generating alerts accordingly. The host state is a static tree structure including elements in a fixed hierarchical relationship, the elements being given value by associated tokens, the elements and associated tokens representing the hardware and software components of the computer system. The alert definitions generate alerts according to the values of at least one token, at least one alert or a combination of various tokens and/or alerts. The host state is created by providing a static tree structure representing a general computer system. Component information indicating hardware and software components of the computer system is extracted from diagnostic data of the computer system. The host state is generated according to the static tree structure and the component information.

20 Claims, 15 Drawing Sheets



U.S. PATENT DOCUMENTS

5,699,505	12/1997	Srinivasan	395/183.07
5,726,912	3/1998	Krall, Jr. et al.	364/550
5,727,144	3/1998	Brady et al.	395/182.04
5,751,964	5/1998	Ordanic et al.	395/200.54
5,758,071	5/1998	Burgess et al.	395/200.5
5,825,944 *	10/1998	Wang	382/309
5,908,471	6/1999	Lach et al.	714/805
5,909,540	6/1999	Carter et al.	395/182.02
5,944,839	8/1999	Isenberg	714/26

* cited by examiner

1/15

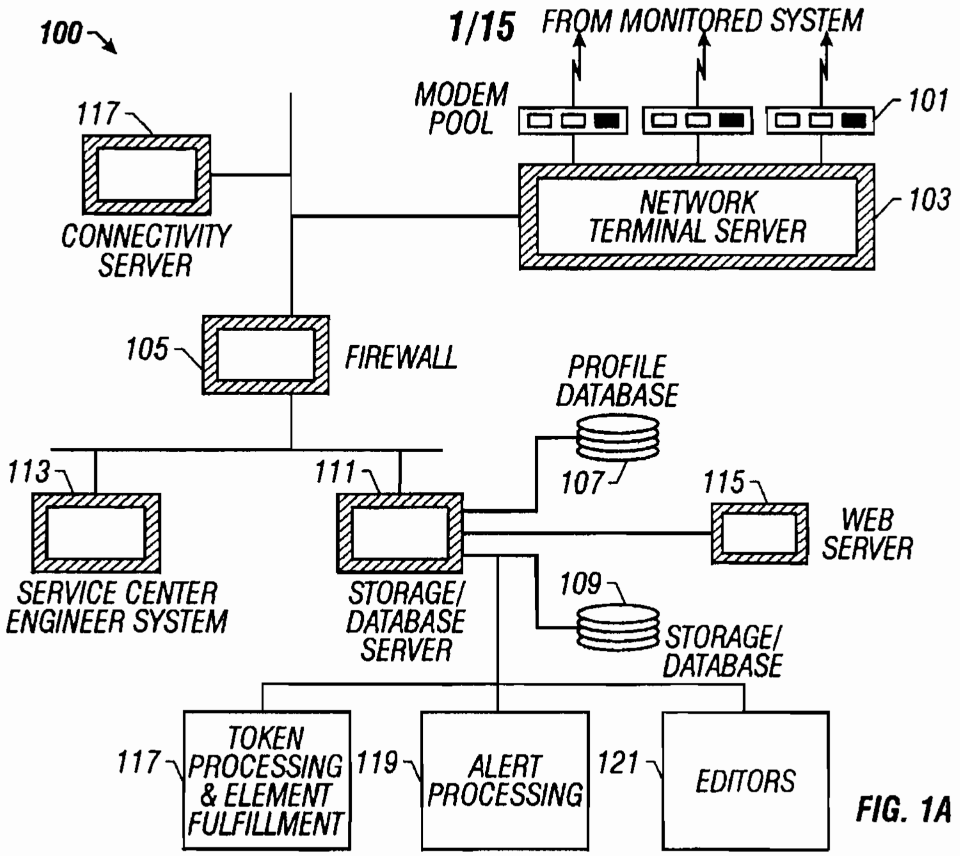


FIG. 1A

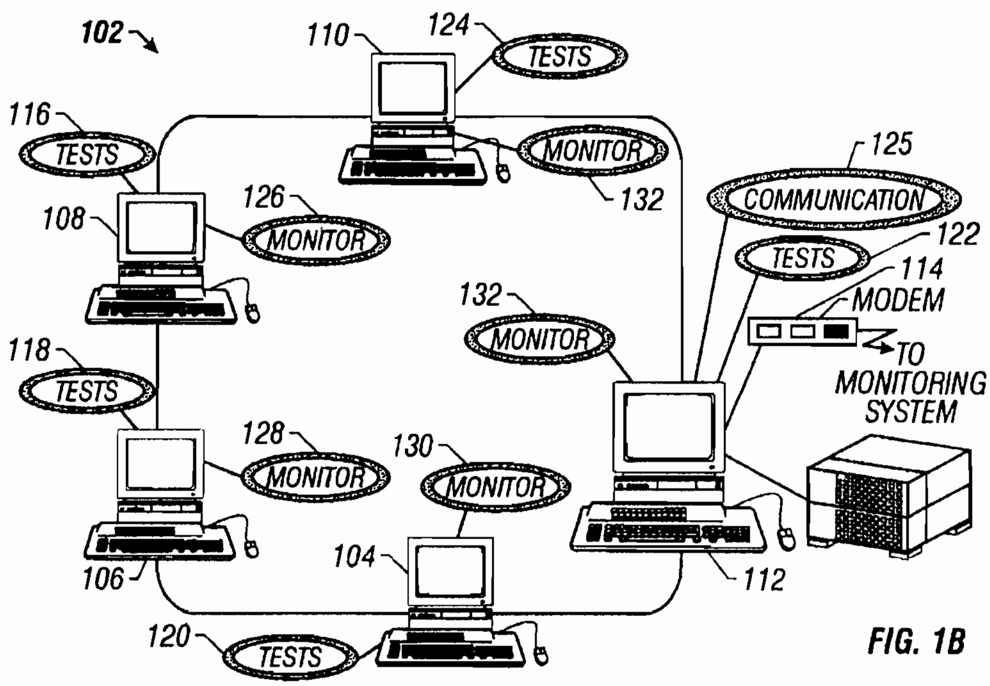


FIG. 1B

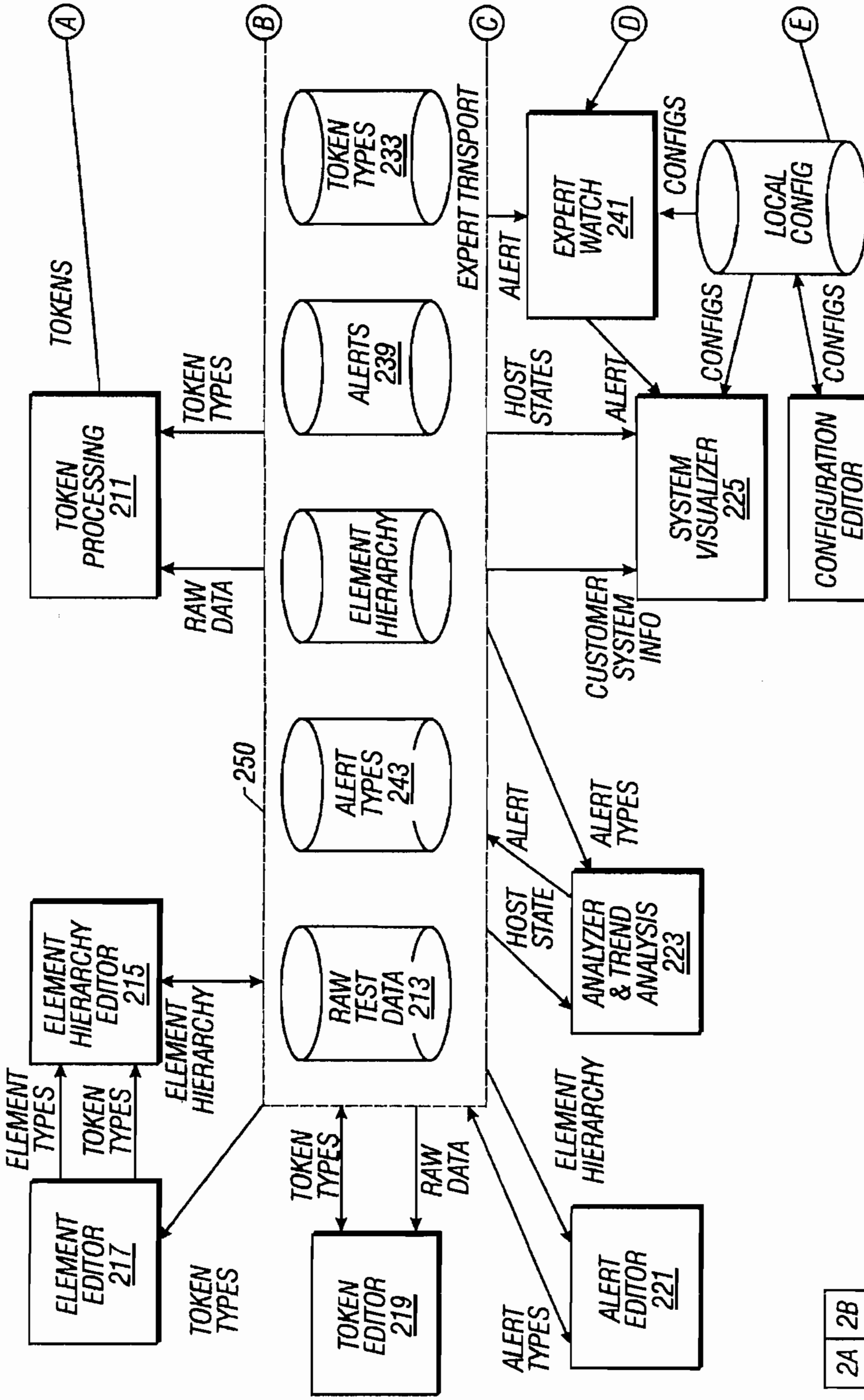

TOKEN PROCESSING 211
TOKENS
TOKEN TYPES
RAW DATA
TOKEN TYPES 233
ALERTS 239
ELEMENT HIERARCHY
ALERT TYPES 243
RAW TEST DATA 213
250
EXPERT TRNSPORT
ALERT
EXPERT WATCH 241
CONFIGS
LOCAL CONFIG
CONFIGS
CONFIGS
HOST STATES
ALERT
SYSTEM VISUALIZER 225
CONFIGURATION EDITOR 227
CUSTOMER SYSTEM INFO
ALERT
ALERT TYPES
HOST STATE
ANALYZER & TREND ANALYSIS 223
ELEMENT HIERARCHY EDITOR 215
ELEMENT HIERARCHY
ELEMENT TYPES
TOKEN TYPES
ELEMENT EDITOR 217
TOKEN TYPES
TOKEN TYPES
RAW DATA
TOKEN EDITOR 219
ELEMENT HIERARCHY
ALERT TYPES
ALERT EDITOR 221
2A 2B

FIG. 2A

FIG. 2

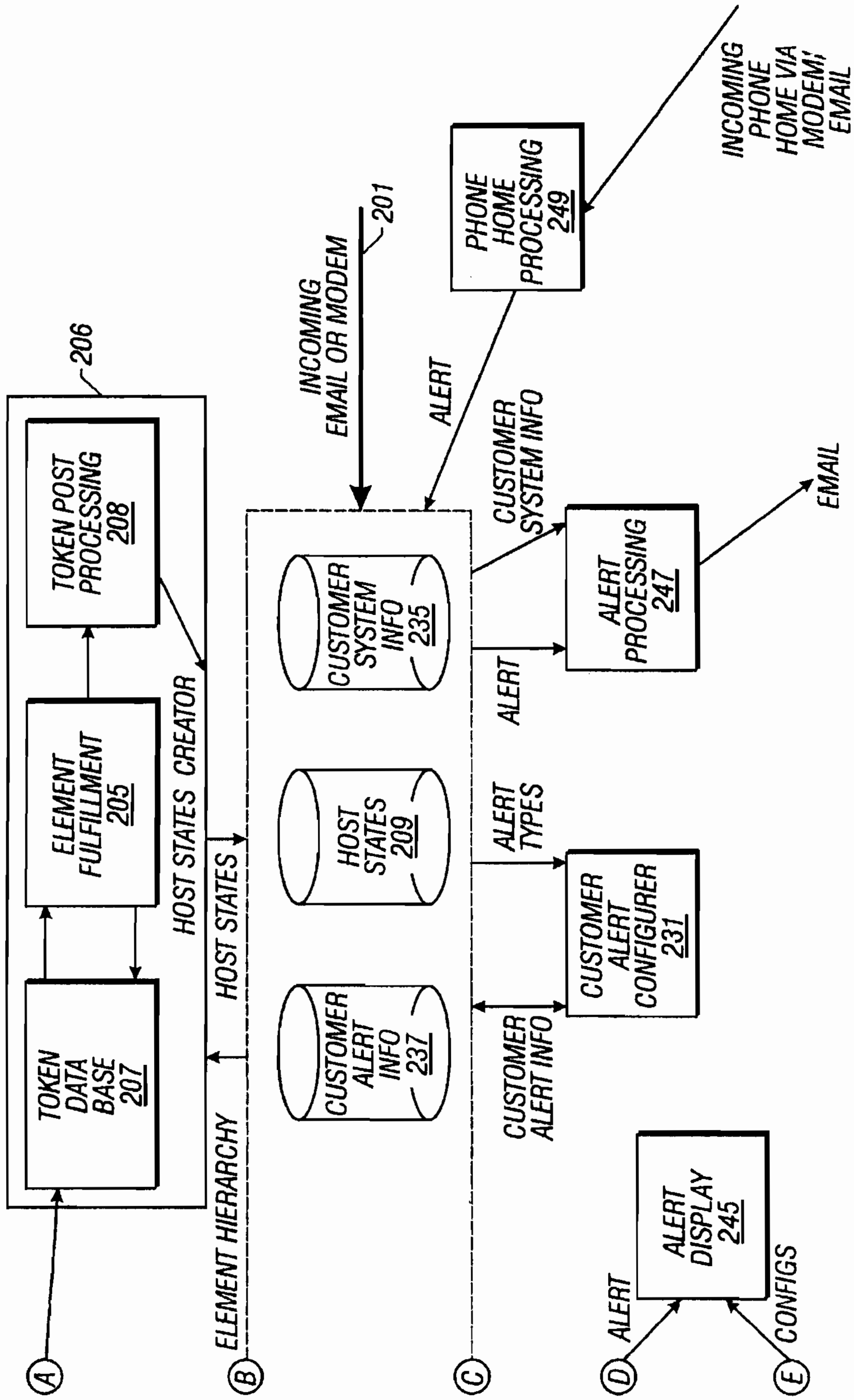

206
TOKEN DATA BASE 207
ELEMENT FULFILLMENT 205
TOKEN POST PROCESSING 208
HOST STATES CREATOR
HOST STATES
ELEMENT HIERARCHY
INCOMING EMAIL OR MODEM
201
PHONE HOME PROCESSING 249
INCOMING PHONE HOME VIA MODEM; EMAIL
ALERT
CUSTOMER ALERT INFO 237
HOST STATES 209
CUSTOMER SYSTEM INFO 235
CUSTOMER SYSTEM INFO
ALERT PROCESSING 247
EMAIL
ALERT
ALERT TYPES
CUSTOMER ALERT INFO
CUSTOMER ALERT CONFIGURER 231
ALERT DISPLAY 245
ALERT
CONFIGS
A
B
C
D
E

FIG. 2B

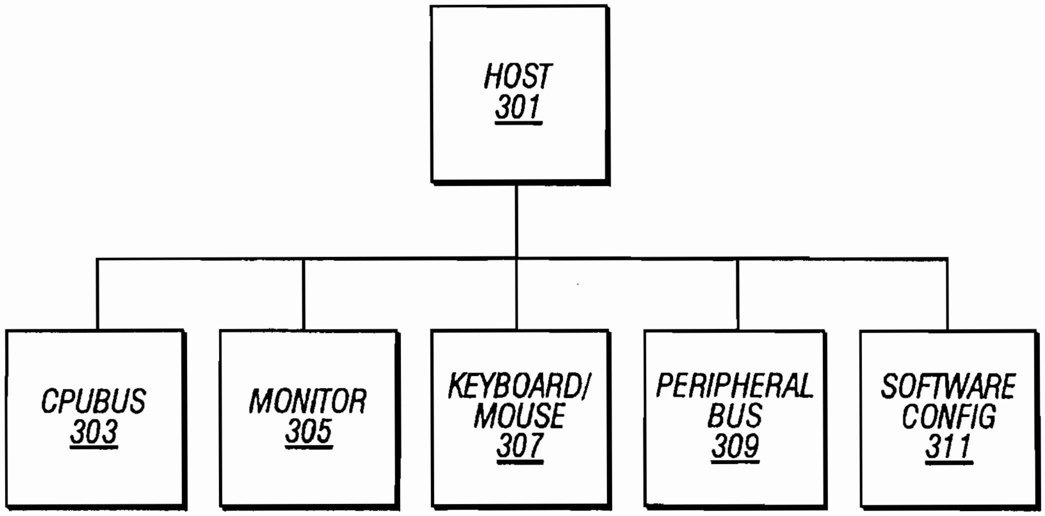

HOST
301
CPUBUS
303
MONITOR
305
KEYBOARD/
MOUSE
307
PERIPHERAL
BUS
309
SOFTWARE
CONFIG
311

FIG. 3

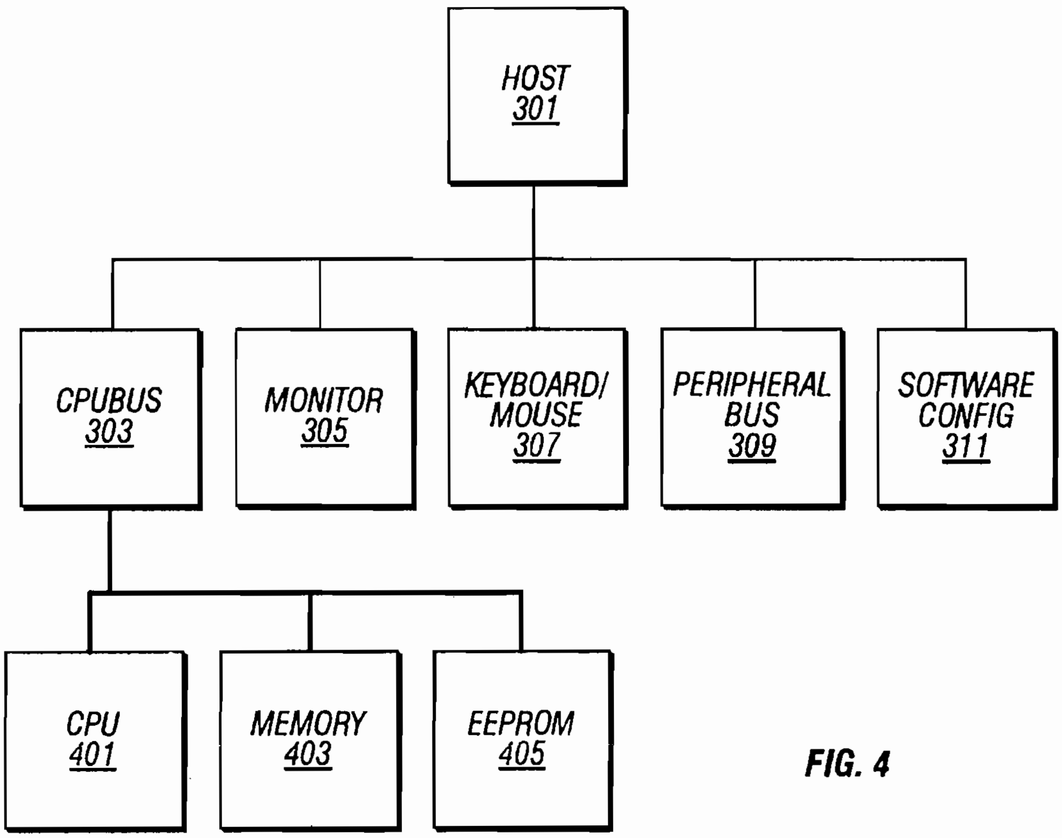

HOST
301
CPUBUS
303
MONITOR
305
KEYBOARD/
MOUSE
307
PERIPHERAL
BUS
309
SOFTWARE
CONFIG
311
CPU
401
MEMORY
403
EEPROM
405

FIG. 4

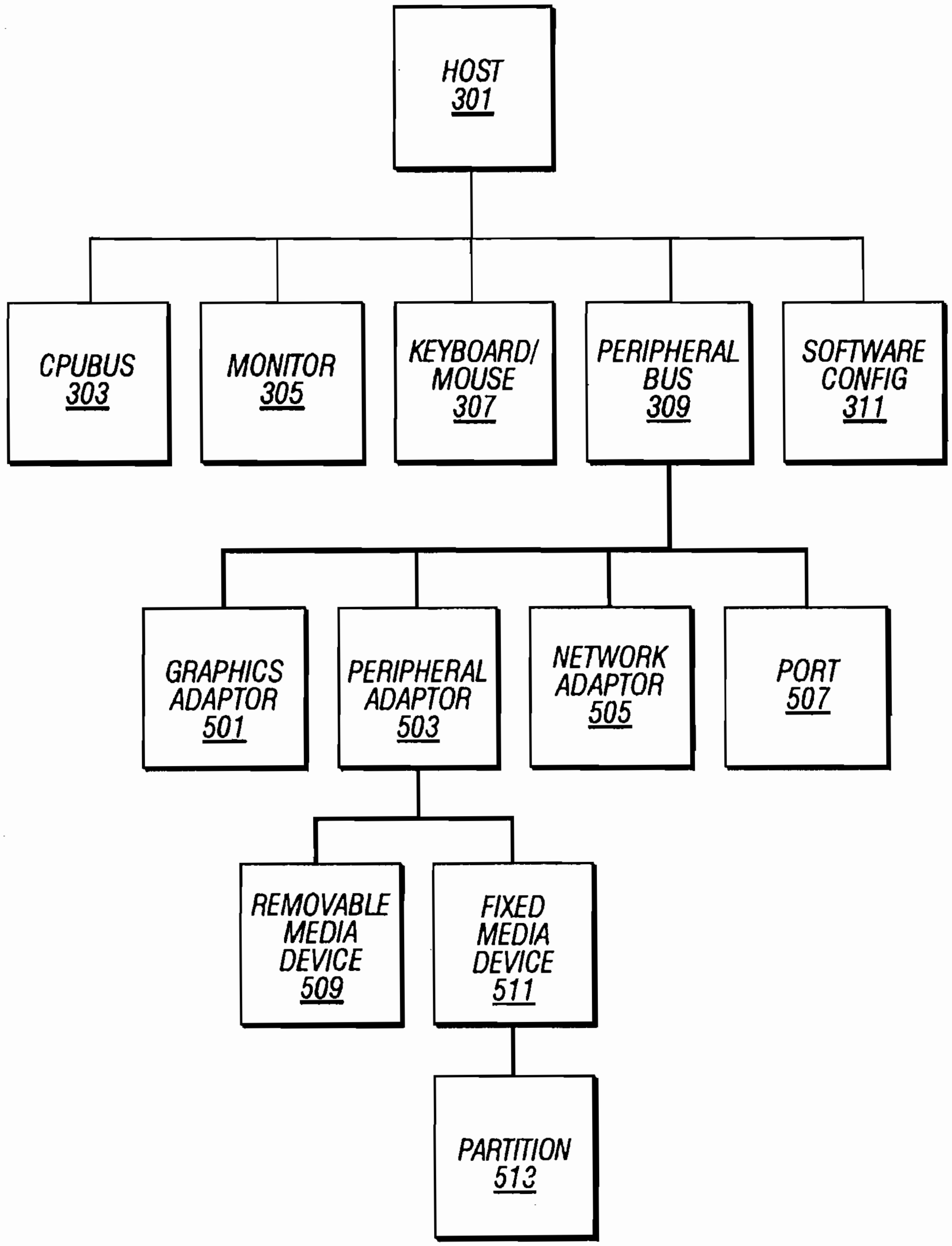

HOST
301
CPUBUS
303
MONITOR
305
KEYBOARD/
MOUSE
307
PERIPHERAL
BUS
309
SOFTWARE
CONFIG
311
GRAPHICS
ADAPTOR
501
PERIPHERAL
ADAPTOR
503
NETWORK
ADAPTOR
505
PORT
507
REMOVABLE
MEDIA
DEVICE
509
FIXED
MEDIA
DEVICE
511
PARTITION
513

FIG. 5

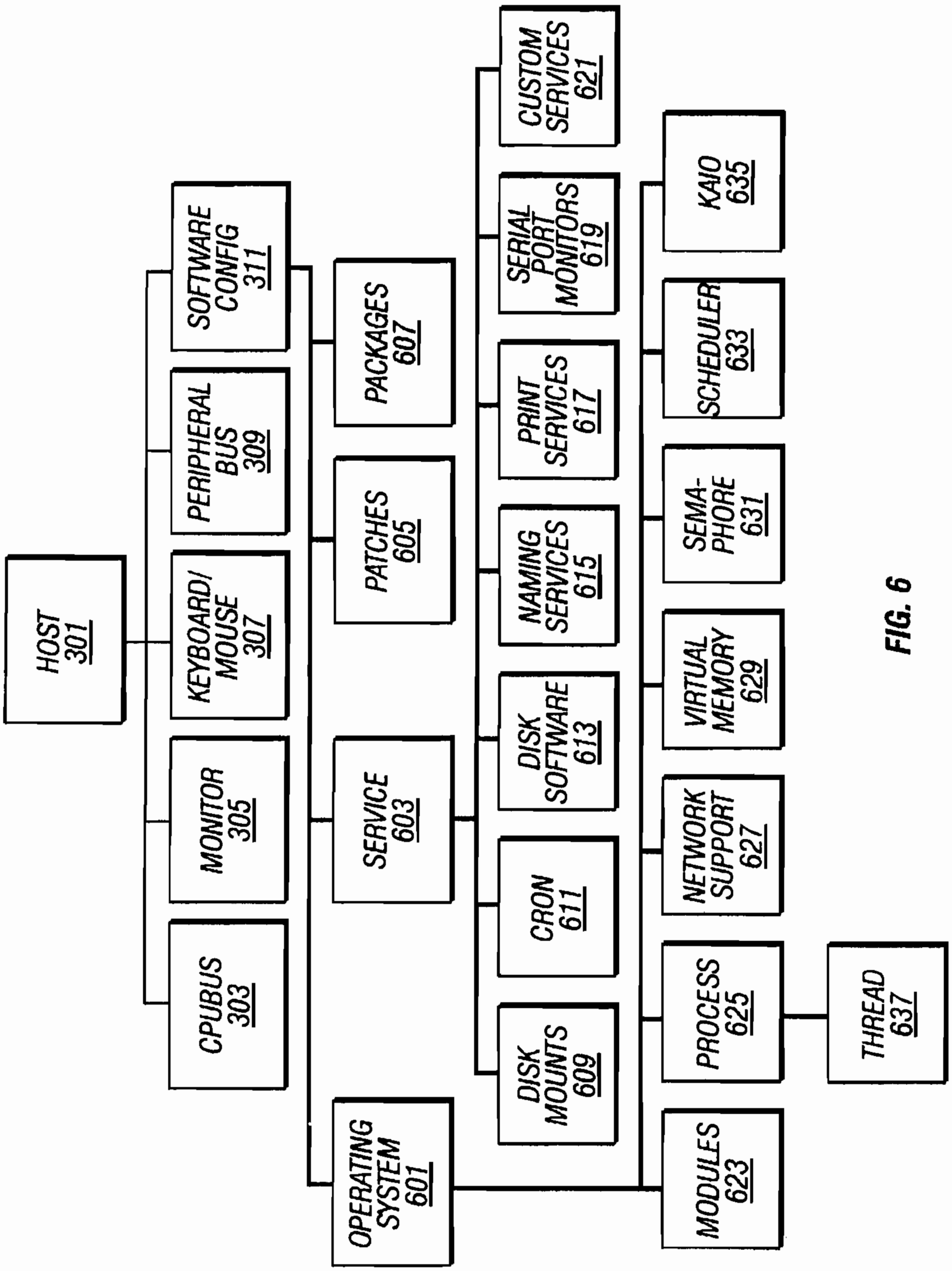

HOST 301
CPUBUS 303
MONITOR 305
KEYBOARD/ MOUSE 307
PERIPHERAL BUS 309
SOFTWARE CONFIG 311
OPERATING SYSTEM 601
SERVICE 603
PATCHES 605
PACKAGES 607
DISK MOUNTS 609
CRON 611
DISK SOFTWARE 613
NAMING SERVICES 615
PRINT SERVICES 617
SERIAL PORT MONITORS 619
CUSTOM SERVICES 621
MODULES 623
PROCESS 625
NETWORK SUPPORT 627
VIRTUAL MEMORY 629
SEMA-PHORE 631
SCHEDULER 633
KAIO 635
THREAD 637

FIG. 6

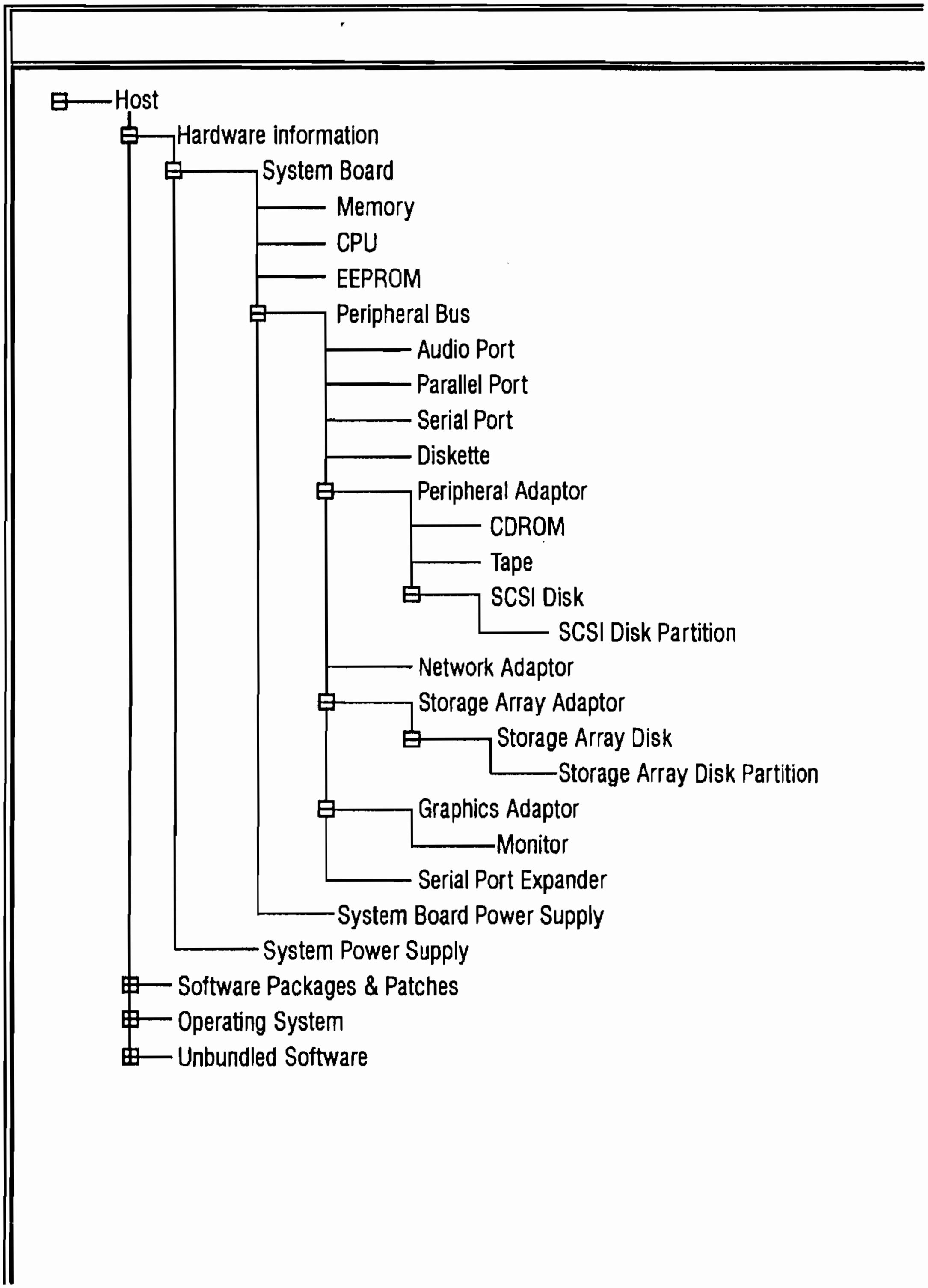

Host
Hardware information
System Board
Memory
CPU
EEPROM
Peripheral Bus
Audio Port
Parallel Port
Serial Port
Diskette
Peripheral Adaptor
CDROM
Tape
SCSI Disk
SCSI Disk Partition
Network Adaptor
Storage Array Adaptor
Storage Array Disk
Storage Array Disk Partition
Graphics Adaptor
Monitor
Serial Port Expander
System Board Power Supply
System Power Supply
Software Packages & Patches
Operating System
Unbundled Software

FIG. 7A

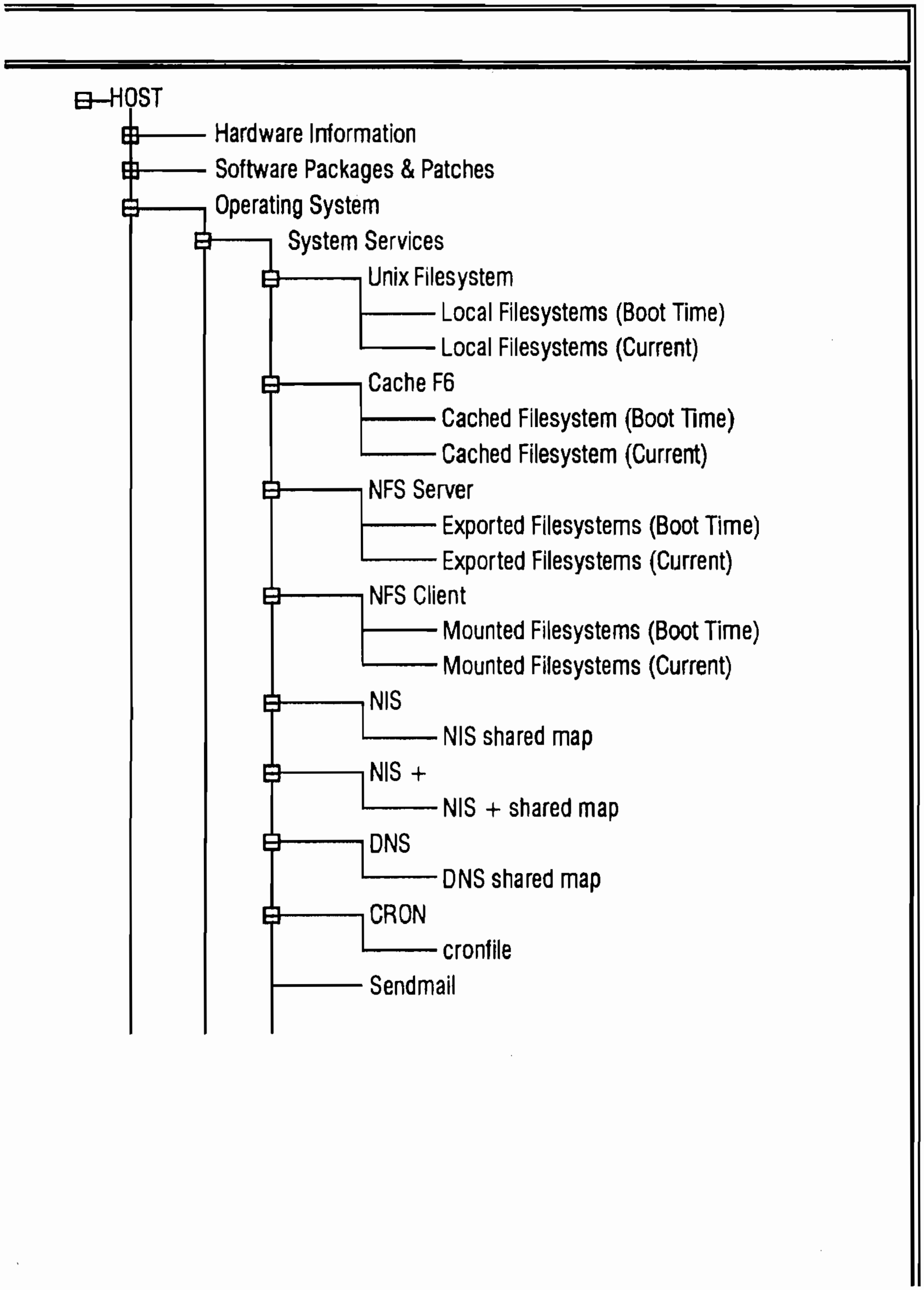

HOST
Hardware Information
Software Packages & Patches
Operating System
System Services
Unix Filesystem
Local Filesystems (Boot Time)
Local Filesystems (Current)
Cache F6
Cached Filesystem (Boot Time)
Cached Filesystem (Current)
NFS Server
Exported Filesystems (Boot Time)
Exported Filesystems (Current)
NFS Client
Mounted Filesystems (Boot Time)
Mounted Filesystems (Current)
NIS
NIS shared map
NIS +
NIS + shared map
DNS
DNS shared map
CRON
cronfile
Sendmail

FIG. 7B

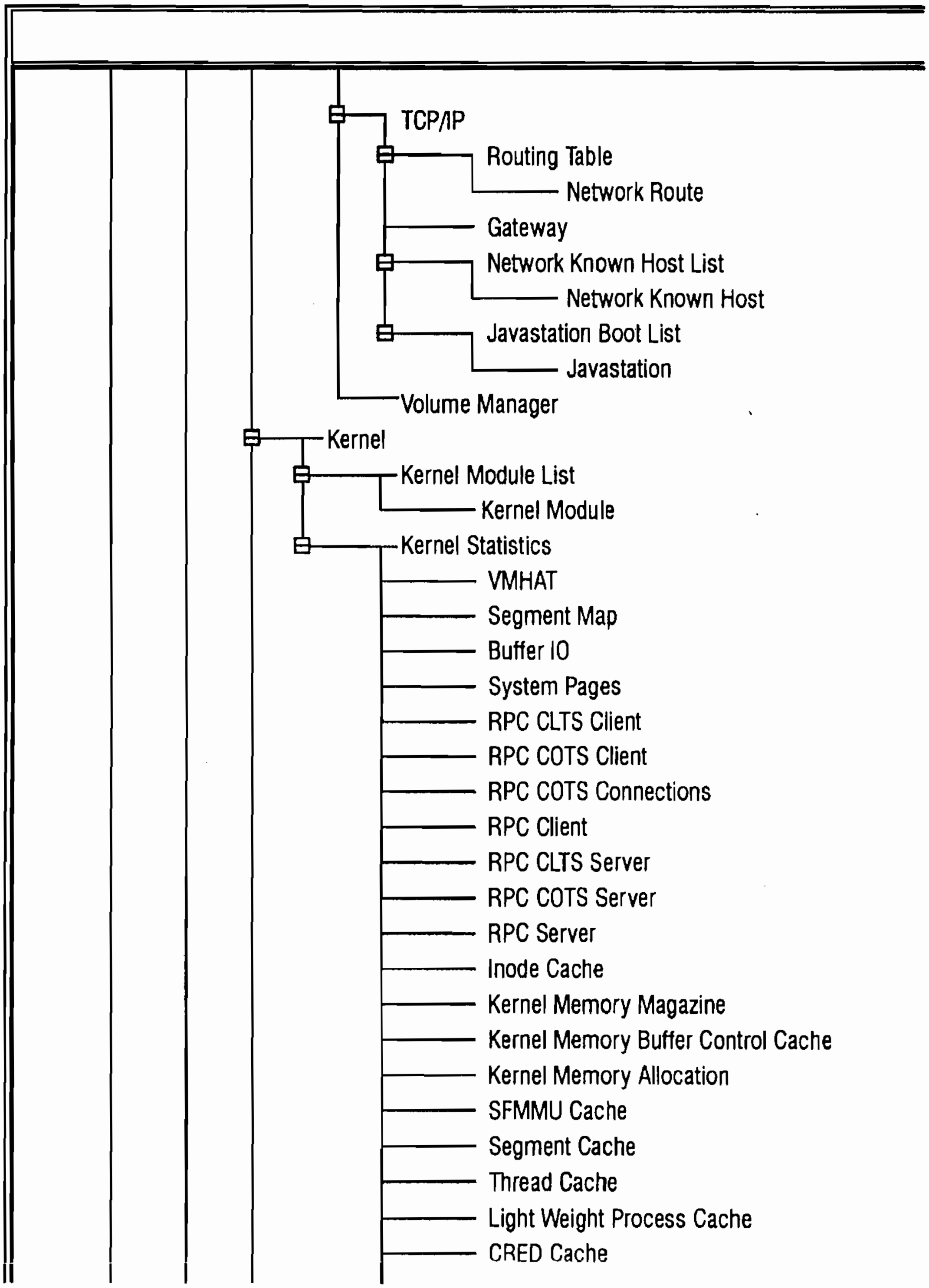

TCP/IP
Routing Table
Network Route
Gateway
Network Known Host List
Network Known Host
Javastation Boot List
Javastation
Volume Manager
Kernel
Kernel Module List
Kernel Module
Kernel Statistics
VMHAT
Segment Map
Buffer IO
System Pages
RPC CLTS Client
RPC COTS Client
RPC COTS Connections
RPC Client
RPC CLTS Server
RPC COTS Server
RPC Server
Inode Cache
Kernel Memory Magazine
Kernel Memory Buffer Control Cache
Kernel Memory Allocation
SFMMU Cache
Segment Cache
Thread Cache
Light Weight Process Cache
CRED Cache

FIG. 7C

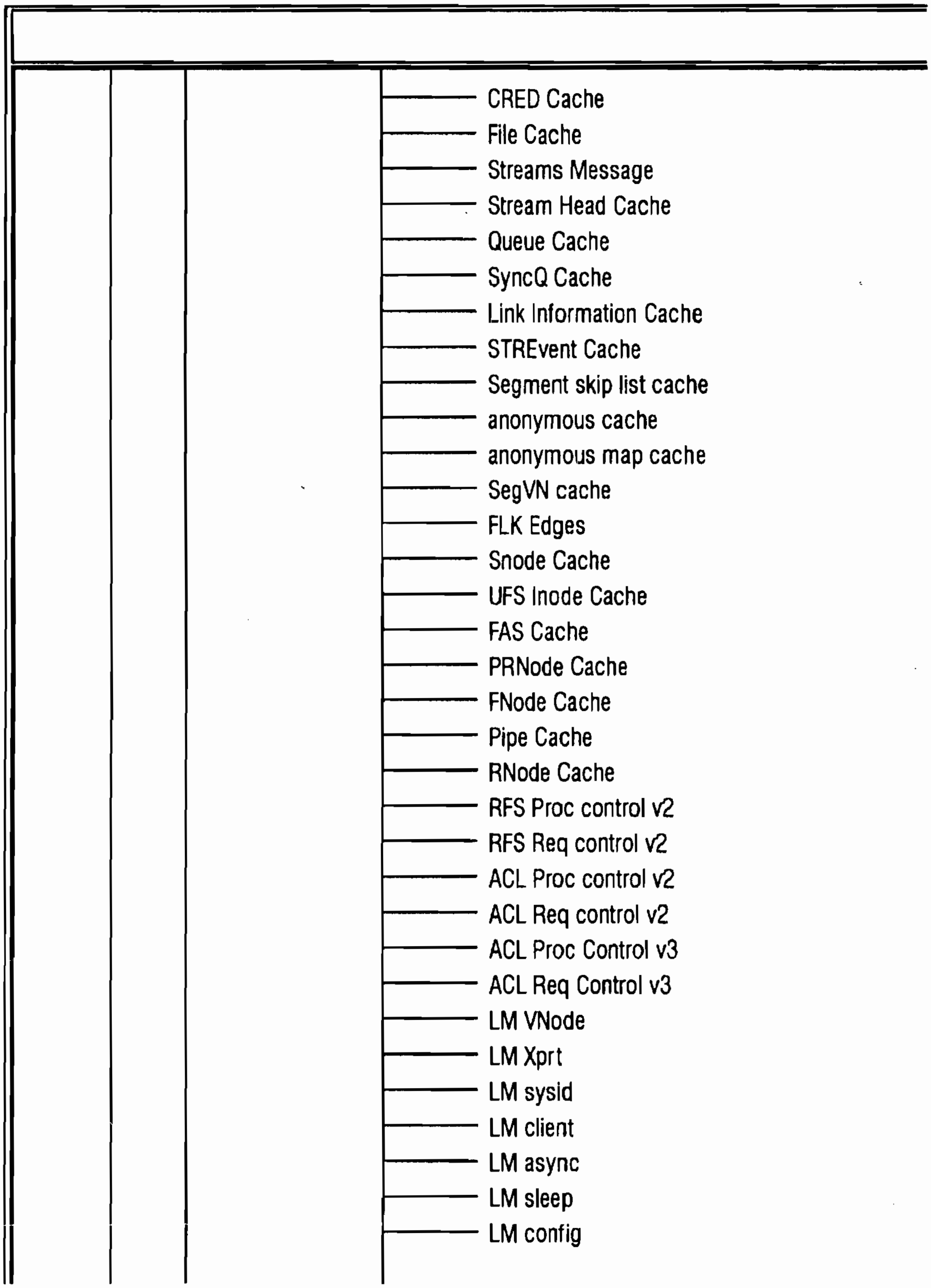

CRED Cache
File Cache
Streams Message
Stream Head Cache
Queue Cache
SyncQ Cache
Link Information Cache
STREvent Cache
Segment skip list cache
anonymous cache
anonymous map cache
SegVN cache
FLK Edges
Snode Cache
UFS Inode Cache
FAS Cache
PRNode Cache
FNode Cache
Pipe Cache
RNode Cache
RFS Proc control v2
RFS Req control v2
ACL Proc control v2
ACL Req control v2
ACL Proc Control v3
ACL Req Control v3
LM VNode
LM Xprt
LM sysid
LM client
LM async
LM sleep
LM config

FIG. 7D

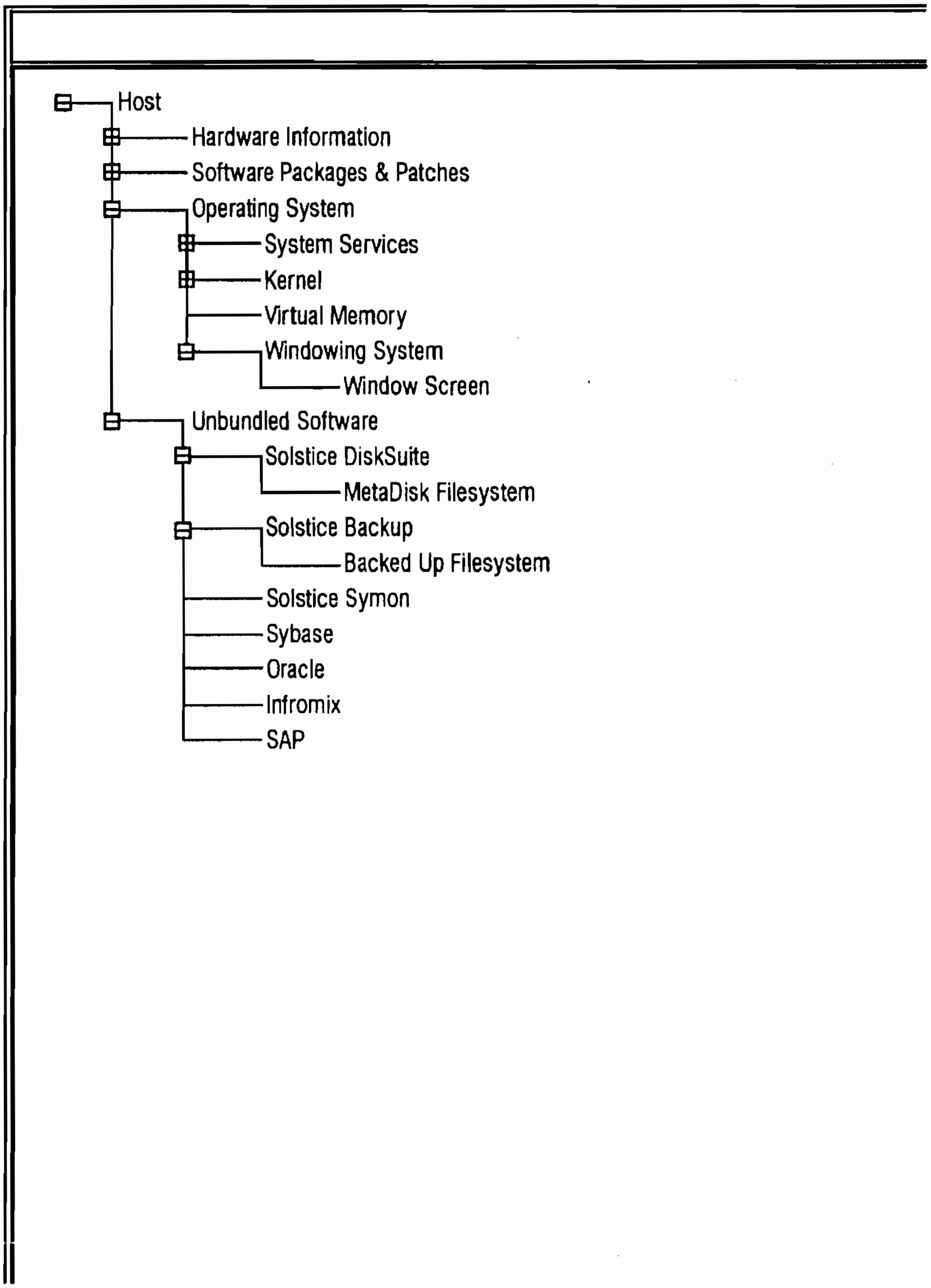

Host
Hardware Information
Software Packages & Patches
Operating System
System Services
Kernel
Virtual Memory
Windowing System
Window Screen
Unbundled Software
Solstice DiskSuite
MetaDisk Filesystem
Solstice Backup
Backed Up Filesystem
Solstice Symon
Sybase
Oracle
Infromix
SAP

FIG. 7E

* Partition	Tag	Flags	Sector	Count	Sector	Mount Directory
c0t0d0s0	2	00	0	2048960	2048959	/
c0t0d0s1	3	01	2048960	262960	2311919	
c0t0d0s2	5	00	0	4154160	4154159	
c0t0d0s7	8	00	2311920	1842240	4154159	/export/home

* Partition	Tag	Flags	Sector	Count	Sector	Mount Directory
c0t1d0s0	2	00	0	62320	62319	/c0t1d0s0.a005WN
c0t1d0s1	3	01	62320	197600	259913	
c0t1d0s2	5	01	0	4154160	4154159	
c0t1d0s0	4	00	259920	3894240	4154159	/c0t1d0s6.a005WN

FIG. 8

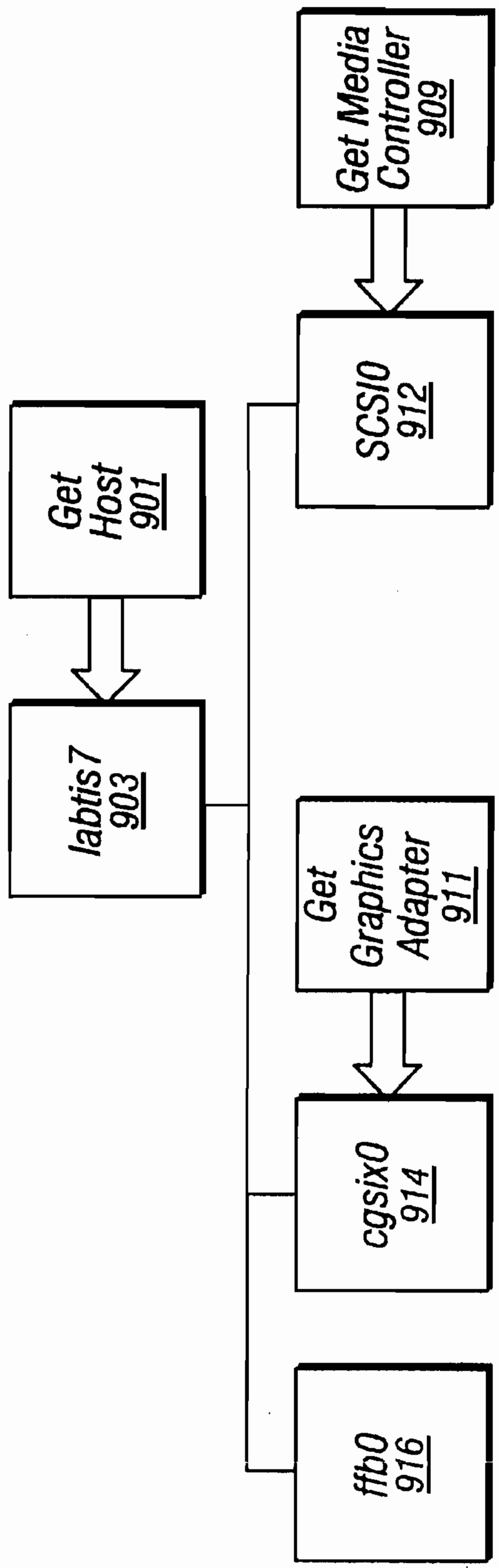

Get Media Controller 909
SCSI0 912
Get Host 901
labtis7 903
Get Graphics Adapter 911
cgsix0 914
ffb0 916

FIG. 9

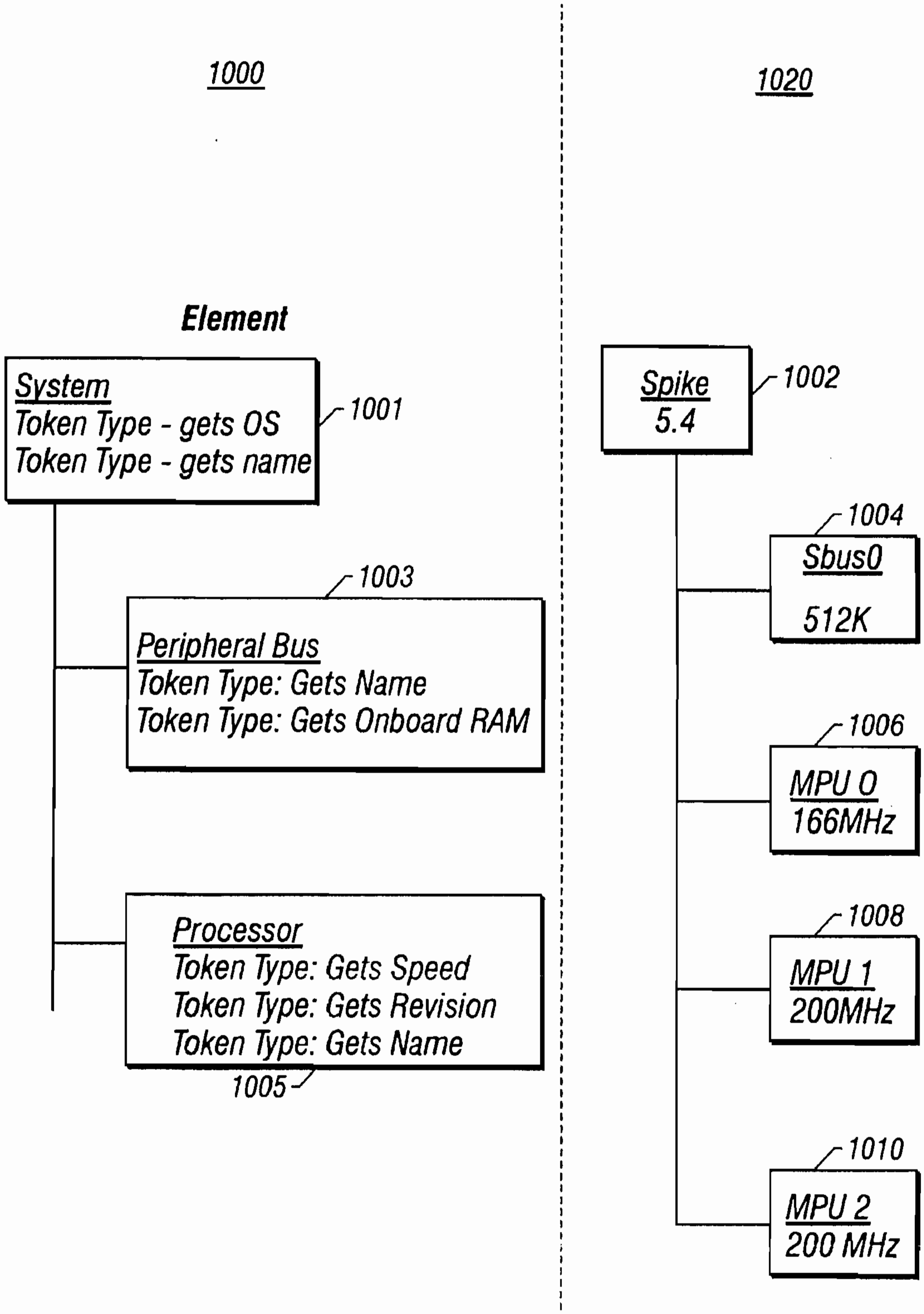

1000
1020
Element
System
Token Type - gets OS
Token Type - gets name
1001
Spike
5.4
1002
1003
Peripheral Bus
Token Type: Gets Name
Token Type: Gets Onboard RAM
1004
Sbus0
512K
1006
MPU 0
166MHz
Processor
Token Type: Gets Speed
Token Type: Gets Revision
Token Type: Gets Name
1005
1008
MPU 1
200MHz
1010
MPU 2
200 MHz

FIG. 10

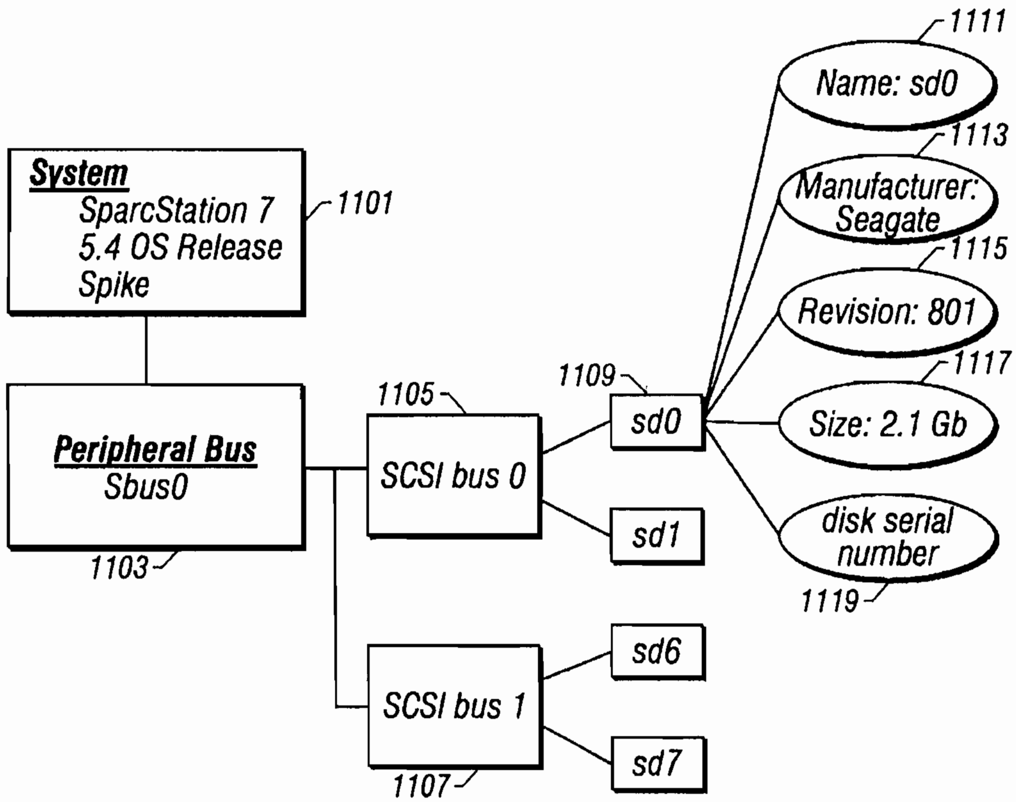

System
SparcStation 7
5.4 OS Release
Spike
1101
Peripheral Bus
Sbus0
1103
1105
SCSI bus 0
1107
SCSI bus 1
1109
sd0
sd1
sd6
sd7
1111
Name: sd0
1113
Manufacturer: Seagate
1115
Revision: 801
1117
Size: 2.1 Gb
disk serial number
1119

FIG. 11

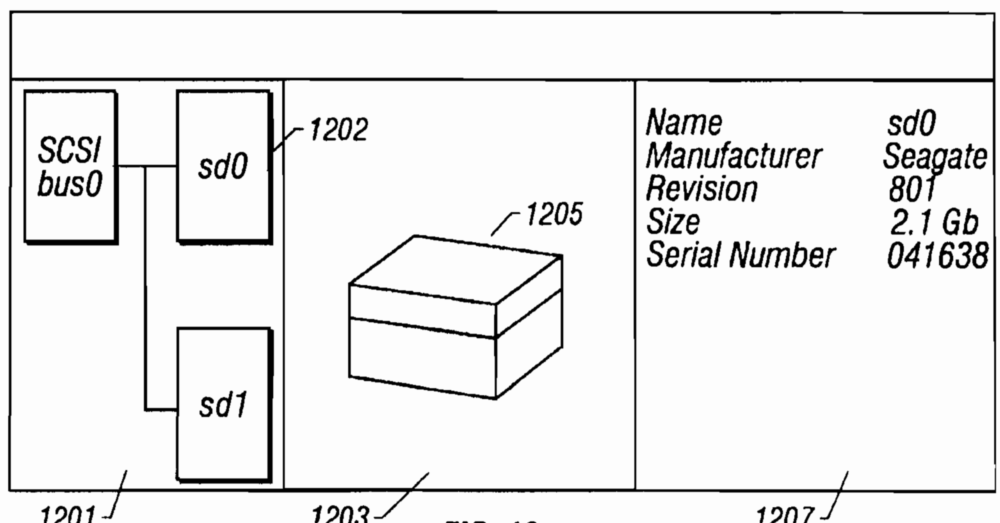

SCSI bus0
sd0
sd1
1202
1205
Name sd0
Manufacturer Seagate
Revision 801
Size 2.1 Gb
Serial Number 041638
1201
1203
1207

FIG. 12

REMOTE ALERT MONITORING AND TREND ANALYSIS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application relates to the following commonly owned co-pending applications, Ser. No. 08/819,501, entitled "AUTOMATIC REMOTE COMPUTER MONITORING SYSTEM", by Michael J. Wookey, filed Mar. 17, 1997, and Ser. No. 08/819,500, entitled "DYNAMIC TEST UPDATE IN A REMOTE COMPUTER MONITORING SYSTEM", by Michael J. Wookey, filed Mar. 17, 1997, Ser. No. 08/829,276, entitled "REBUILDING COMPUTER STATES REMOTELY", by Michael J. Wookey, filed Mar. 31, 1997, which applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to monitoring of computer systems and more particularly to monitoring the state of a computer system.

2. Description of the Related Art

Computer systems such as mainframes, minicomputers, workstations and personal computers, experience hardware and software failures that degrade system performance or render the system inoperative. In order to diagnose such failures computer systems include diagnostic capability which provides various types of system diagnostic information.

Computer systems are typically serviced when a failure is noticed either by system diagnostics or by users of the system when the system become partially or completely inoperative. Since computer systems are frequently located at some distance from the support engineers, when problems do occur, a support engineer may access the computer system remotely through a modem in an interactive manner to evaluate the state of the computer system. That remote dial-in approach does allow the support engineer to provide assistance to a remote customer without the delay of traveling to the computer system site. Once connected to the remote computer system, the support engineer can perform such tasks as analyzing hardware and software faults by checking patch status, analyzing messages file, checking configurations of add-on hardware, unbundled software, and networking products, uploading patches to the customer system in emergency situations, helping with problematic installs of additional software, running on-line diagnostics to help analyze hardware failures, copying files to or from customer system as needed.

However, there are limitations to such support. For instance, the data size transfer may be limited at the time of failure, due to such factors as modem speed and thus a complete picture of a system may be unavailable. Running diagnostic software during the remote session, if necessary, may adversely impact system performance. Where a system is part of a network, which is commonplace today, the running of diagnostic tests may impact network performance. Where computer systems are being used in a production or other realtime environment, such degradation of system performance is obviously undesirable.

Further, historical data on system performance is not be available in such scenarios. It is therefore impossible to analyze trends or compare system performance, e.g., before and after a new hardware of software change was made to the system. The support engineer is limited to the snapshot of the system based on the diagnostic information available when the support engineer dials in to the system.

It would be advantageous if a support engineer had available complete diagnostic information rather than just a snapshot, However, system diagnostic tests typically generate a significant amount of data and it can be difficult for a support engineer to analyze such data in a raw form. Additionally, service centers typically support a number of different computer systems. Each computer system has its own hardware and software components and thus have unique problems. For example, it is not uncommon for failures to be caused by incorrect or incompatible configuration of the various hardware and/or software components of the particular system. It would be advantageous to provide a remote monitoring diagnostic system that could process, present and manipulate diagnostic data in a structure and organized form and also monitor a number of different computer systems without having prior knowledge of the particular hardware or software configuration of each system being monitored. In order to provide better diagnostic support to computer systems, it would also be advantageous to provide the ability to detect problems in the diagnostic data and to provide proactive monitoring of the diagnostic data in order to better detect and/or predict system problems.

SUMMARY OF THE INVENTION

Accordingly, the present invention provides a method, apparatus and computer program products to generate alerts indicating predetermined conditions exist in a computer system. In one embodiment in accordance with the present invention, the method includes providing a host state representing a state of the computer system, comparing alert definitions to the host state to determine if conditions defined in the alert definitions exist in the host state; and generating alerts in response to the comparing of alert definitions. The host state is a static tree structure including elements in a fixed hierarchical relationship, the elements being given value by associated tokens, the elements and associated tokens representing the hardware and software components of the computer system. The alert definitions generate alerts according to the values of at least one token, at least one alert or a combination of various tokens and/or alerts. The host state is created by providing a static tree structure representing a general computer system. Component information indicating hardware and software components of the computer system is extracted from diagnostic data of the computer system. The host state is generated according to the static tree structure and the component information. The static tree structure includes element types in a fixed hierarchical relationship and the element types represent the hardware and software components of the computer system.

In another embodiment in accordance with the present invention, a monitoring computer system apparatus for generating alerts indicating predetermined conditions exist in a monitored computer system, includes a first data storage area storing a plurality of alert definitions defining respective predetermined conditions in the monitored computer system. A second data storage area stores at least a first host state of the monitored computer system, the first host state having associated token values indicating respective of software and hardware components of the monitored computer system. A monitoring computer is coupled to the first and second data storage areas and the monitoring computer generates alerts when a condition defined in one of the alert definitions is determined to be present in the first host state.

The method, apparatus and computer program products of the present invention provide a component based data structure for the diagnostic data that facilitates problem detection as well as proactive monitoring of the monitored computer system. Further, the present invention can build a representation of a monitored computer system without having any prior knowledge of the hardware and software details of the monitored computer system. Further, the invention can provide support for new computer systems and products in a manner that is more responsive than was previously available.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings, wherein the use of the same reference symbols in different drawings indicates similar or identical items.

FIG. 1*a* shows an exemplary system for rebuilding the state of a computer according to the present invention.

FIG. 1*b* shows an exemplary monitored computer system which runs diagnostic tests on each computer and communicates the results of those tests to the system of FIG. 1*a*.

FIG. **2** details the architecture of a system that rebuilds computer states according to the present invention.

FIG. **3** shows a root and lower branches of a static tree definition of computer system.

FIG. **4** shows additional branches of a static tree definition of a computer system relates to components of the CPU-BUS.

FIG. **5** shows additional branches of a static tree definition of a computer system, related to components on the peripheral bus.

FIG. **6** shows additional branches of a static tree definition of a computer system, related to software configuration components.

FIG. 7*a* shows the root and lower branches of a second exemplary tree structure.

FIG. 7*b* shows additional sub elements of the System services element.

FIG. 7*c* shows additional operating system elements.

FIG. 7*d* shows operating system elements related to kernel statistics.

FIG. 7*e* shows unbundled software elements.

FIG. **8** shows an exemplary output of a diagnostic test from which tokens are extracted and used to instantiate the static model exemplified by FIGS. **3–6** and FIGS. 7*a*–7*e*.

FIG. **9** shows an exemplary instantiation of a portion of a static tree.

FIG. **10** shows another example of a tree structure and an instantiation of that tree.

FIG. **11** shows another example of a host state.

FIG. **12** shows how the host state can be displayed to show graphical, and attribute information about the host state.

DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

Referring to FIGS. 1*a* and 1*b*, an exemplary computer system **100**, according to the present invention, receives diagnostic data from a monitored computer system **102**. Monitored computer system **102** runs diagnostic tests, from among tests such as those shown in Table 1 or Table 2, on a periodic basis. The monitored system includes at least one computer and typically includes a plurality of computers **104**, **106**, **108**, **110**, and **112** coupled in a network as shown in FIG. 1*b*. The diagnostic tests **116**, **118**, **120**, **122**, and **124** are run on the computer system **102** under the control of monitor control software **126**, **128**, **130**, **132**, and **134**. The results of those diagnostic tests are automatically provided at periodic intervals to the computer system **100** which monitors computer system **102**. In exemplary computer system **100**, which includes one or more computers and associated storage areas, preferably coupled in a network, incoming diagnostic data from monitored system **102** is received from modem **114** at one of the modems in the modem pool **101**. The incoming data may be received via email or may be a direct modem connection to the monitored system **102** or may be received via other communication channels. The raw diagnostic data is stored in storage **109**. Storage **109** is shown as a single storage unit but may be separate storage units to accommodate the various storage requirements described herein. In order to perform operations on the data received, processor **117** transforms the received incoming data into a structure which can then be analyzed by alert processing computer **119**. Editing capability is provided by a separate computer **121**. Note that the functions may be performed in separate machines or may be combined into one or several computers.

TABLE 1

Class	Test Name	Description
network	automount.files	Automount/etc Files
	automount.nis+	Automount NIS+ Files
	automount.nis	Automount NIS Files
	dfshares	NFS shared filesystems
	domainname	Domain name
	etc.defaultdomain	/etc/defaultdomain
	etc.defaultrouter	/etc/defaultrouter
	etc.dfstab	List/etc/dfs/dfstab
	etc.hostnames	/etc/hostname(s)
	etc.hosts	/etc/hosts
	etc.mnttab	List/etc/mnttab
	etc.named.boot	/etc/named.boot
	etc.nsswitch.conf	/etc/nsswitch.conf
	etc.resolv.conf	/etc/resolv.conf
	netstat-an	List all TCP connecfions
	netstat-in	List network interfaces
	netstat-k	Network interface low-level statistics
	netstat-rn	List network routing
	nisdefaults	NIS+ server defaults
	nisstat	NIS+ statistics
	ypwhich	NIS server name
	ypwhich-m	NIS map information
OS	checkcore	Check for core files
	df	Disk Usage
	dmesg	Boot Messages
	framebuffer	Default console/framebuffer
	hostid	Numeric ID of host
	ifconfig	Ethernet/IP configuration
	messages	System messages (/var/adm/messages)
	patches	List system patches
	pkginfo	Software package information
	prtconf	System hardware configuration (Software Nodes)
	prtconf-p	System hardware configuration (PROM Nodes)
	prtdiag	Print diagnostics (Sun-4d systems only)
	sar	System activity reporter
	share	Shared directories
	showrev	Machine and software revision information
	swap	Swap report
	uptime	Local uptime and load average

TABLE 1-continued

Class	Test Name	Description
	whatami	Lengthy system description report
unbundled	fddi-nf_stat	FDDI low-level statistics
	metastat	Online DiskSuite or Solstice DiskSuite
	vxprint	Systems using SPARCstorage Array Volume Manager
	x25_stat	X.25 low-level statistics

TABLE 2

Test Name	Test Name
ps -ef	ypwhich
pkginfo -1	df
vmstat	df -k
showrev -a	mount -v
xdpyinfo	more/etc/dfs/dfstab
netstat -k	cachefsstat
kmemleak (SMCC)	df -1
vtsprobe	df -1k
modinfo	showrev -p
arp -a	nettest -1v (VTS)
netstat -r	dmesg
configd	diskprobe
more/etc/mail/sendmail.cf	disktest -1v (VTS)
crontab -1 (as root)	tapetest -1v (VTS)
more/etc/nsswitch.conf	bpptest -1v (VTS)
more/etc/resolv.conf	uname -a
niscat -o org_dir	

Referring to FIG. 2, the architecture of a system according to the present invention, is shown in greater detail. Incoming diagnostic data **201** is received via email or direct modem link (or another communication link) into the monitoring system and stored in raw test data storage area **213**. The test data, which contains information about the software and hardware components in monitored system **102**, is processed by token processing **211** to extract the information associated with hardware and software components in the monitored system. The extracted information is then used to create a representation of the monitored system in host state creator **206** based on the component information. The host state is the state of the monitored system or one computer of the monitored system over the particular time period that the diagnostic tests were run. Further details of the host state will be described further herein.

In order to create a representation of the monitored system, the components contained in the test data are built into a system hierarchy based on a static hierarchy tree definition. In a preferred embodiment, one static hierarchy tree definition is applicable to all systems which are being monitored. The extracted information about the components in the monitored system are mapped onto the static tree to create the system representation for the monitored system. Thus, the state of the monitored system is rebuilt.

The hierarchy tree is composed of elements. An elements can be thought of as a physical or virtual component of a computer system. For example, a computer system may include such components as a disk, a disk partition, a software package, and a patch. An element has tokens associated with it. Thus, a partition element may have a disk percentage token, disk name token, and space available token associated with it. An element definition includes what token types fulfill the element, and give the element value. In one embodiment, an element is an instance of a class of elements types as implemented in an object oriented language such as the JAVA programming language (JAVA™ and JAVA-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.).

An exemplary portion of a static tree definition a computer system is shown in FIGS. 3–6. FIG. **3** shows a lower level (closer to the root) elements of the static tree and FIGS. **4**, **5**, and **6** show how the tree definition expands. The element host **301** defines the kind of computer that is being monitored. For instance, the host may be a Sun workstation running a Solaris™ operating system (Solaris and Sun are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.) or a PC running a Windows NT operating system. Attached to hose **201** are other physical or virtual components such as CPU bus **303**, monitor **305**, keyboard/mouse **307**, peripheral bus **309** and software configuration **311**. Note that the terms are very general. Each element represents types of components that can be found in a typical computer system.

Referring to FIG. **4**, the computer system further includes additional physical or virtual components on the CPU bus **303**. The additional elements found on the CPU bus include CPU **401**, memory **403** and EEProm **405**. Referring to FIG. **5**, additional components of the static hierarchy tree definition of the computer system can be found under peripheral bus element **309**. Note that the instance of the peripheral bus could be an Sbus. However, the instance could also be a Peripheral Component Interface (PCI) bus. In fact there could be two instances of peripheral bus, e.g. SBUS and PCF bus. In some instances there could be more than two peripheral buses. The additional elements found on peripheral bus **309** include display adapter **501**, peripheral adapter **503**, network adapter **505** and port **507**. The peripheral adapter element **503** may be coupled to additional elements such as removable media device elements **509**, (e.g., a disk drive, tape or CD drive) or a fixed media device **511**. The fixed media device may be a hard disk drive which can have a further virtual component, partition elements **513**. Note the general nature of the static hierarchy system definition. That allows the static definition to be used even for monitored systems that utilize different software and hardware components.

Referring to FIG. **6**, additional software elements under the software configuration element **311** are shown. Included in the software configuration **311** are the operating system element **601**, software services element **603**, patches element **605** and packages element **607**. Additional elements under software services include disk mounts **609**, cron **611**, disk software **613**, naming services **615**, print services **617**, serial port monitors **619** and custom services **621**. The packages elements **607** indicate, e.g., what software has been installed on the system. The operating system **601** is further defined by elements **623**–**637**.

The description of the static tree is exemplary. Another tree may be chosen according to the system being monitored. Additionally, the static tree may be modified to reflect hardware and software enhancements to computer systems. The hierarchy tree definition is static in that it does not vary according to the system being monitored. However, the hierarchy tree can be edited in element hierarchy editor **215** to accommodate additions and/or deletions from the hierarchy tree when for instance, a new technology begins to be utilized in the monitored computer systems. One static tree or hierarchy tree definition may be sufficient for most or all monitored systems. However, a hierarchy tree definition could be tailored to the type of computer system that is being monitored to e.g., enhance processing speed. Another exem-

plary tree structure is shown in FIGS. 7a–7e. The tree structure can be seen to include both hardware components and software components.

Thus, given a static definition of a generic computer system such as shown in FIGS. 3–6, or FIGS. 7a–7e. it is possible to build a representation of the actual computer system being monitored utilizing the diagnostic data communicated from the monitored system to the monitoring system.

In order to extract information from the diagnostic data stream, "token types" are utilized. A token type defines each token to have a token name and a test name. A test name comes from the tests shown e.g., in Table 1 or in Table 2, and indicates which test output contains the information for the token. In addition to a token name and a test name, each token has a label and a value. The label for the token gives the token knowledge about what element the token is associated with, i.e., the parent of the token which is an element. The value of the token provides a value extracted from the diagnostic data that gives value to the element.

For instance, assume a disk element exists with a name of "c0t10d0". Assume also that a token exists for such a disk element indicating the number of sectors per cylinder. The name of such a token would be, e.g., "number of sectors per cylinder." The test name in the token would be "vtsprobe" since the output of that test provides the information needed for the number of sectors per cylinder. The label for the token would be "c0t10d0" indicating that token is associated with a particular disk having that name. Finally, the token would have a value which indicates the number of sectors per cylinder. Other tokens could of course be associated with that element. For example, another token associated with that disk element might be a disk manufacturer token that identifies the manufacturer as "Seagate". The value of the token in such an instance would be "Seagate".

Note that one token type can create many tokens from the test data. For example, a "disk name" token type could extract multiple tokens, e.g. the disk names "c0t1d0" and "c0t2d0", from the test data when a particular system has two disks so named.

There are two types of tokens. The first is an element realizing token. Element realizing tokens provide a way to determine whether an element should be included when building a particular host state. For example, a disk name token is an element realizing token. The second type of token are data tokens which provide additional information about an element that has already been realized, such as the token indicating the number of sector per cylinder. Thus, it can be seen that tokens give value to the elements.

For any particular system, it is preferable to create tokens with as much granularity as possible. Thus, the smallest piece of information that is available about a system from the available diagnostic tests should be included as a token. Representative tokens are included in the description herein. The exact nature of the tokens and the total number of tokens will depend upon the system that is being monitored, including its hardware and operating system, and the diagnostic tests that can be run on the system. Table 3, attached, shows both elements and tokens for an exemplary embodiment of the invention. For each element shown in Table 3, the associated tokens are shown as well as the tests that supply the token information. In addition Table 3 shows the types of computers and operating system releases on which the tests are operable.

An exemplary output of one the diagnostic tests is shown in FIG. 8. The processing must extract from the output such information as the disk partition ID, last sector, first sector and the like. Examples of the tokens that are extracted for disk partition elements is shown in Table 3 for tokens associated with "SCSI Disk Partition Element". In order to parse through the output of the diagnostic tests a strong textual processing programming language, such as Perl, is utilized.

Note that the preferred implementation of the invention described herein is in an object oriented computer language and more particularly in JAVA. Nearly all the classes and type definitions described herein extend the type Persistent Object. Persistence is a technique that can be used in object oriented programming to ensure that all memory resident information can be stored to disk at any time. It can be through of as encoding and decoding. When a persistent object is saved to disk, It is encoded in some manner so that it may be efficiently stored in the appropriate medium. Equally when loading the information back, it is decoded. That allows complex memory structures to be stored easily in databases with minimum disk space impact.

Now that it is understood that a static tree structure is composed of elements which are realized and given value by tokens, the building of a particular representation of a monitored computer system can be more completely described. Referring again to FIG. 2, the incoming data stream **201** of diagnostic data is stored in raw test data storage area **213**. Token types are stored in storage area **233**. The token types and the diagnostic data are provided to token processing **211**, which is the process of running the token definitions against the incoming data and generating an outgoing stream of tokens which are stored in token data base **207**. In a preferred embodiment the tokens in token data base **207** are stored as a hashtable to provide faster access to subsequent processing steps of building the representation of the system. A hashtable is a common key/element pair storage mechanism. Thus, for the token hashtable, the key to access a location in the hashtable is the token name and the element of the key/element pair would be the token value. Note that because the diagnostic data may include data for multiple computers in a monitored network or subnetwork, one task is to separate the diagnostic data provided to the token processing process **211** according to the computer on which the diagnostic tests were executed. Token types are run against the test output indicated in the test name in the token. For example token types having a test name parameter of "df" are run against "df" test output.

Once all the raw test data has been processed and a completed token data base **207** is available, the second set of processing operations to build the representation of the monitored computer may be completed. In order to understand the building of the tree, an examination of several typical features of an element class will provide insight into how an element is used to build a tree.

An element has methods to retrieve the name of the element as well as the various values associated with an element. For example, a disk element includes a method to retrieve a disk ID token which realizes the element as well as having a method to find in the token data base a disk capacity parameter, sectors per track and other tokens such as those shown in Table 3 associated with "SCSI Disk". Those parameters are used to realize a disk element and give it value.

An element of one type is similar to an element of another type. For example, a partition element requires different tokens to provide different values but otherwise is similar to a disk element. The tokens needed to provide value to the

partition element may include partition size, partitions used and partition free. Note elements have associated tokens providing a name or ID. As previously described, tokens have both a value and a label. The label or name provides a "tie" for the token. Suppose a disk element is instanced with a name of "c0t1d0". One of its token to be fulfilled is disk size. The token that provides the disk size would have a name of "c0t1d0" and a value of 1.2 Gb. The value of 1.2 Gb would be tied to the name "c0t1d0".

Referring to FIG. **9**, an example of building a host state based on the elements of the static tree is shown. The term "host state" refers to the representation of the monitored system based on its diagnostic data. The host state essentially describes the state of a system for a given time period. The host state may be viewed as an instantiated element hierarchy based on the raw data that has come in from the remote host. In other words, it is a completed element hierarchy with value. The diagnostic data is collected over a particular time period, so the host state represents the state of the monitored machine over that particular time period, e.g., an hour. The host state is built by starting from the top of the tree element host **301**. The element **301** has methods to retrieve relevant tokens from the token data base **207**. As shown in FIG. **9**, the element **301** is realized with Get Host **901** as "labtis 7" **903**. Because the token data base is a hashtable in the preferred embodiment, the realization of each element is faster. Next element graphics adapter **501** gets **(911)** graphics adapter cgsix**0** **914** and ffb**0** **916**. Continuing to build the host state, media controller element gets **(909)** SCSI**0** **912** from the data base. In a preferred embodiment, the host state is built in depth order meaning that each element and all branches of that element are built before another element is built. Thus, referring back to FIG. **5**, for example, everything on peripheral bus **309** would be built before the building of the software configuration **311**. For each element in the static tree, the token data base is searched and the host state is created in element fulfillment processing **205** which requests tokens from token data base **207** in the form of searches for tokens providing realization and value to the static tree.

Once the element fulfillment stage is completed a final token post processing operation takes place in **208**. An element can have a token defined that is the mathematical result of other tokens. For example, a disk space free token is derived from a simple subtraction from a disk used token and a total disk space token. The calculations are completed in this post processing operation **208** to complete the host state.

Note that because the tree definition is static and is intended to be general, not all elements will be found in every host state. Thus, when building the host state, no data will be found in the token data base for a particular element that is lacking in the monitored system. Additionally, in some host states, an element will be found more than once. Thus, the tree structure provides the flexibility to build host states that look very different.

Once the host state is built, it is saved in host state storage **209**. The storage of the host state provides several advantages. For one, it provides the capability to search back through time and to compare one host state with another host state from a different time or perform trend analysis over time. The host states may be stored for nay amount of time for which adequate storage area is available. For example, host states may be stored for a year.

Additionally, the stored host states are used when the diagnostic data is incomplete. There may be occasions when a test has failed to run in the monitored system or has not run before a scheduled communication of data from the monitored system. That may cause problems in the building of the host state from the static tree, especially where the test was one that created elements lower in the tree (i.e. towards the root). Each element can include a value that indicates how critical the element is to the system. If the element is critical, such as a disk, there could be a problem with the system and it should be noticed. If the data is not critical to the system, then older data could be retrieved from the previous host state in time for that particular host. That could be limited by restricting such retrieval to a specified number of times, e.g., 10, or any other number appropriate to the criticality of the element, before marking data as invalid.

Referring again to FIG. **2**, the expert transport **250** provides access to all of the data storage mediums used for the various processes requiring the storage mediums. The communications between processing and storage elements is preferably network based to allow flexibility in implementation as the load of the subsystems may be distributed across machines if need be. Each module can access the expert transport in a very rigid manner making use of the object orientated design facilities provided by JAVA.

A second example of building a host state is shown in FIG. **10**. Element **1001** has associated token types for the name of the system and the OS. Peripheral bus element **1003** has associated token types which gets the name of the peripheral/bus and any onboard RAM. Element **1005**, which is a processor element, has associated token types to provide a name, a revision number and the processor speed. The static definition **1000** creates a host state **1020** where the system is realized as "Spike" with an OS release of 5.4. The peripheral bus is instantiated as Sbus**0** with 512 K of RAM. The processor element is instantiated three times as MPU**0** **1006**, MPU**1** **1008** and MPU**2** **1010**. Thus, an example is provided where a single element is realized more than one time in a particular system.

Referring to FIG. **11**, another example of a host state is provided. The system is shown as element **1101** with associated values of being SparcStation**2**, with a system name Spike and an OS 5.4 release. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc., in the United State and outer countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. The system has a peripheral bus, Sbus**0**, which has two SCSI buses **1105** and **1107**. Attached on SCSI bus **0** are two disks sd**0** and sd**1**. Disk "sd**0**" has associated tokens, in addition to its name, the manufacturer **1113**, the revision **1115**, the size of the disk, **1117** and the serial number **1119**. As seen in Table 3, for the SCSI disk element, other tokens may be associated with a disk element.

In addition to storing the host state in data base **209**, the system provides a graphical interface to access information about the host state. Referring to FIG. **12**, an exemplary system visualization screen is shown. The tree structure is provided in region **1201** of the screen which graphically represents a portion of the host state shown in FIG. **11**. Tree structures may also be represented in the form shown in FIGS. 7*a*–7*e* or other appropriate form. In addition to displaying the tree structure which provides the user a graphical depiction of the completed element hierarchy for a particular system at a particular time, the screen also provides a graphic image of the particular component which is being viewed. For instance, region **1203** of the screen shows a graphic image **1205** of a disk. Assuming that the viewer had clicked on disk **1202**, sd**0**, region **1207** shows the

attributes or token values associated with the selected element. Thus, the attributes relating to name, manufacturer, revision, size and serial number are all provided. This presents the support engineer with an easily understandable graphical image of the total system, and any particular component of the system that is represented in the host state, along with pertinent attributes.

Referring again to FIG. 2, the system visualizer **225** receives host states from host states database **209** and customer system information stored in data base **235**. The system visualizer also receives alerts and local configurations relevant to a particular support engineer. The first task that the system visualizer must be to select the particular host that is to be worked upon or viewed. Thus, the system visualizer will have to search the host states database **209**. The visualizer will provide the ability to parse through time to select from all the host states available for a particular system. While each element may have a graphic associated with it, a separate graphic can be used to indicate that a problem exists with a particular element.

In addition to displaying the attributes of an element, which are the values of the tokens associated with the element, the system visualizer provides graphical capability to graph attributes against time. One or more attributes can be selected to be graphed against history. In other words, the same attributes from different instances of the element hierarchy for a particular system can be compared graphically. For example, the amount of disk free over time can be monitored by looking at outputs of the "dt" test over a period of time. The df output includes such token values as disk percentage used for a particular partition, partition name and size of partition. The visualizer will extract the tokens representing amount of disk percentage used for a particular set of host states. The host states from which the disk percentage tokens are extracted is determined according to the time period to be viewed. That information can then be visualized by plotting a graph of disk percentage used against time. Also, the visualizer can view different instances of the host state. In other words, the visualizer can view the state of a monitored system at different times. That capability provides a visual interpretation of changes in system configuration. The visualizer accesses the stored multiple instances of the host state of the particular system to provide that capability.

While it is possible for the diagnostic data from the monitored system to come up to the monitoring system in a raw form, it is also possible to do some preprocessing on the data in the monitored system. The preprocessing could translate the diagnostic data to something more easily readable by the monitoring system. As a simple example, the monitored system could eliminate all white space in the test output. The choice of whether to do preprocessing may depend on such considerations as whether the additional load put on the monitored system is a cost that is outweighed by the benefit of simple processing at the monitoring system.

Once host states have been created, the data can be analyzed for the presence of alerts. Alerts are predefined conditions in the various components of the monitored computer system that indicate operating conditions within the system. The alerts are designed to be sufficiently flexible so that they can detect not only serious problems, but also detect performance and misconfiguration problems. Different levels of severity may be provided in each alert. For example, alert severity can range from one to six. Severity level six indicates effectively that the system has gone down while severity level of one indicates that here could be a performance problem in the system.

Two types of alerts are available. The first kind of alert is a spot alert which is based on current data only. A spot alert indicates that a particular value of a system component has exceeded a threshold value. For example, a spot alert could result when the number of parity errors exceeds a predetermined threshold, or when the root partition of a disk exceeds 99%. A patch configuration problem provides another example of a spot alert. For example, assume the patch configuration problem exists for a particular patch in a particular OS release. If a host state contains the token indicating the presence of the particular patch as well as the token indicating the particular OS release, an alert would be issued.

The second type of alert is a predictive alert. A predictive alert analyzes historical and current data to identify trends. In other words, the predictive alert is a form of trend analysis. Storing multiple instances of stored host states in the host state data base, makes possible such trend analysis of the operating conditions of a monitored system. Trend analysis allows pro-active detection of undesirable conditions in the collected diagnostic data. For example, trend analysis identifies that the number of memory parity errors is increasing, even though the number is not yet fatal. The alert can generate the probability that the increase will eventually result in a fatal error. Another example of a predictive alert is memory leak detection.

Trend analysis compares the value of a current alert to previous alert results. The trend is determined by comparing, e.g., tokens continuing the number of parity errors of a memory element, over a sequence of host states. Trend analysis may use alerts saved from a previous analysis or may obtain relevant token values from saved host states or may operate on both saved tokens from earlier host states as well as saved alert values.

Note that trend analysis may be utilized to detect a build up of data indicating an increasing number of parity errors over a period of time and can flag the problem before the spot alert was generated. Similarly, the trend analysis can detect increasing disk usage and predict the problem before the threshold of 99% is reached. It can be seen that trend analysis is really analysis performed on the results of spot alerts over time.

A spot alert provides the basic analysis type. The spot alert allows components to be tested against alert types stored in database **243**. Alert types define an alert in a manner similar to a token type defining a token. The alert types define the details of the alert and how to process it. Consider an alert to determine if a particular partition has exceeded a predetermined percentage used. The tokens utilized in processing the alert include a token for the partition name, e.g., /var. A second token utilized is partition percentage used. The alert determines if partition name=/var AND percentage used ≧80%. When those two conditions are true, the alert is raised. That is a simple spot alert.

As an example of a predictive alert consider an alert that predicts whether or not swap space is going to get low on the system. The token value used is one that identifies swap-space used. An operator that is useful in predictive analysis is one called, OverTimeOperator, that provides the value of swap space used over time, i.e., from sequential host states. One can specify how far back the OverTimeOperator should go in retrieving token values from previous host states. The spot test of such a token determines if in the latest data, the swap space used is over 90%. That is the first gating factor of the alert. Then the alert uses that spot test data and the data from the OverTimeOperator and provides the data to a

normalization function which provides a graphical analysis of the data. If the angle of normalization is greater than 52 degrees, an alert is generated thereby predicting that swap space is going to get low on the system. The particular angle selected as a trigger may depend on such factors as the system being monitored and the normalization function.

An exemplary alert definition is shown below which detects a probable swap space problem. In the example, the "OverTimeOperator" retrieves the swap spaced used tokens for the last 48 hours. The swap space used token are retrieved into var1 which is a vector or list of all swap spaced used tokens. Var2 is a vector of vectors which includes var1. Var2 is provided because in one embodiment, the compare operator may operate on more than two things. The result determines if swap spaced used tokens have been greater than 90% over the last 48 hours.

```
Vector var1=OverTimeOperator.dbGet ("token:Swap
   Used", currentTime, current Time—48*3600);
// input for var2
Vector var2input0=new Vector ( );
var2input0.addElement (var1);
Integer var2=((Integer) var2Input0);
Integer var0=new Integer ("constant:int 90");
AlertRes res=GreaterThanOperator.compare (var2, var0);
```

In one embodiment, the alert definitions are run against the host states using alert functions. The code for each alert definition is not actually stored in the Alert function. Instead, the JAVA code for the alert definition is sent by the alert editor to a file repository, e.g., **243** from the compiler. A reference to the compiled alert definition is then stored in the Alert Function which is stored in a database, e.g. database **109** as shown in FIG. **1**. An exemplary AlertFunction class is shown below.

```
Class AlertFunction
{
    String AlertFunction           // reference to actual javacode
    String Name;
```

-continued

```
    Vector CustomersApplicable;    // vector of customers Alert
                                   // function is run on. If
                                   // Empty run on all
    Weight wgt;                    // tells it what the values
                                   // of the function output mean
}
```

Thus, an Alertfunction object will exist for each alert definition, the object pointing to the location where the alert definition actually is stored. The Alertfunction object will be run against the host state (or states) as appropriate.

In one embodiment, there are five possible output severitys, red, yellow, blue, black, green. Weight creates a range mapping onto some or all of these severitys. For instance, if a particular alert returns a number between 1 and 100, a level of between 1 and 20 could be mapped onto red. Similarly, for an alert that returns a value of true or false, a true value can be mapped onto, e.g., red. For each new host state, the alert processor retrieves all of the alert functions. Each alert function points to the associated compiled alert code and in this way all of the alert definitions are parsed against the host state.

When alerts are created, that is when the alert definitions pointed to by the alert functions, are found to exist in a particular host state(s), then an alert object in accordance with an alert class is created. An exemplary alert class is as follows:

```
public class Alert
extends NamedObject
implements Cloneable, Persistence, DatabaseDefinition {
Alert Status      status;           // red,blue,green,yellow
ElemementDef      elementDef;       // eg disk, cpu
Element           element;          // instance of element
AlertFunction     function;         // the function that compute this
                                    // alert, eg check swap space
boolean           isHandled;        // anyone acknowledged it?
ExpertUser        user;             // who acknowledged it
String            soNumber;         // service order # if one was
                                    // logged by RX
String            date;
String            description;      // human readable description, filled
                                    // in from a printf type template
Customer          customer_id;      // uniquely identifies customer site
String            customerOrgName;  // company etc
String            customerSite;     // company etc
CustomerHost      customerHost;     // the specific host
String            customerContact   // name of a person, usually a sys admin
String            customerPhoneNo;  // that person's phone number
int severity;                       // severity level
}
```

Each of the fields above are filled in by either the output value of the AlertFunction or information relevant to the customer that is obtained from the incoming diagnostic data.

Alert types use the element hierarchy as their base and can be tied to the tree definition for visualization purposes. For instance, if an alert is generated for a disk capacity of a partition, the alert visualizer would graphically represent the partition to facilitate ease of understanding for the service engineer.

In a preferred embodiment, alert definitions are processed on each host state after it is generated. Each alert type is compared to a host state and an output is generated. That is, the tokens contained in the host state are compared to the condition defined in the alert type. An alert editor **221** allows alert types to be defined through an editor. An alert, which is an instantiation of a particular alert type, can have an associated severity level as previously described.

An alert may be based on other alerts. That is, an alert type can take either the input from one or more token types or a mixture of other alerts and token types. Therefore a complex alert structure can created before a final alert value is determined. An alert editor **221** provides the ability to create alert types. The alert editor can create the JAVA code to represent the alerts. If the alert type is a fairly rigid structure, the creation of JAVA code is facilitated.

The alert types are related to the element hierarchy. The alert type to test the disk capacity of a partition, as described previously, utilizes tokens related to the partition element in the element hierarchy. That alert works fine for all partitions. In accordance with the model discussed in the element and element hierarchy, only one alert would exist for all partitions created, so all partitions that exist on all disks would have the alert processed when a host state is created.

The alert types, as can be seen from the description of alerts herein, support basic logic tests. As another example, consider an overall test of virtual memory. That may require a disk space alert run on the /tmp partition. For example there may be a /tmp disk space alert, that would be defined upon the global partition, to specify this the alert type would have a logic test to see if the attached token parameter was equal to "/tmp".

There are various operators which are utilized to define the alerts. The operators are in the general sense functions that operate on the token types contained in the host states. Exemplary operators include logical operators, AND, OR, NOT, XOR, BIT-AND, BIT-OR, BIT-NOT, BIT-XOR, arithmetic operators, SUM, SUBTRACT, MULTIPLY, DIVIDE, relational operators, LESS THAN, LESS THAN OR EQUAL, GREATER THAN, GREATER THAN OR EQUAL, EQUALS, NOT EQUALS. There are also set operators, UNION, INTERSECTION, ELEMENT OF, (element of is checking if the particular value is an element of a set), DIFFERENCE BETWEEN 2 SETS. String operators include, STRING LENGTH, STRING-SUBSTRING (to see if the string you have is actually a substring of the original string), STRING-TOKEN, (to see if this particular string is a token of the bigger string). Conversion operators convert, HEXADECIMAL TO DECIMAL, HEXADECIMAL TO OCTAL, HEXADECIMAL TO BINARY. Additional operators are, AVERAGE, MEAN, STANDARD DEVIATION, PERCENTAGE CHANGE, SLOPE (which is based on graphing a straight line interpolation of plots), SECOND ORDER SLOPE, CURVE EXPONENT (map an exponent algorithm on the actual curve), MAX, and MIN, for the maximum and minimum value, ALL OF TYPE (extracts all the values of a certain type out of a host state), ALL OVER TIME (obtains a range of data for a token over a period of time), EXIST, (checks to see if token exists), WEIGHT, (applies a certain weight to a value), NORMALIZE. Some embodiments may also provide for custom operators. Other operators may be utilized in addition to or in place of those described above.

Once the alerts have been defined and stored in alert types database **243**, the alerts have to be run against the host states. Whenever a host state is crated the alert and trend analysis is run against the host state. Thus, the alert types and a host state are provided to analyzer **223**. The analyzer processes the alerts by running the JAVA code definition of the alerts against the host state(s). The alert types may be associated with particular elements so that an entire tree structure does not have to be searched for each alert type. If an alert is generated, alert data base **239** stores the value of the alert. Storing the alerts in a database allows for later retrieval.

Alerts can focus on several major areas of a system operations. Typical areas of interest include patch management, performance monitoring, hardware revision, resource maintenance, software problems, general configurations and hardware failures. Patch management alerts detect if patches are missing on systems that require the patch to correct known hardware or software problems. Performance monitoring and system configuration alerts ensure that the system is configured appropriately to maximize performance. Hardware revision alerts detect when hardware is out of date or a known problem exists with a particular hardware revision. Resource maintenance, e.g., alerts related to swap space, identify when a resource is going to or has run low. Software failure alerts identify known symptoms of software failures. General configuration errors identify system configuration errors that can adversely affect system performance. In addition, hardware failures are also an area of focus for alerts.

In one embodiment of the invention, all alert types are global in that the alert types are run against all monitored systems, i.e., the host state representation of that system, in a default mode. However, the tests can be selectively enabled (or disabled) according to the monitored system. Such capability is provided in the embodiment shown in customer alert configurer **231** which, in a preferred embodiment, is a JAVA based GUI which provides the ability to select which alerts should run on particular monitored systems from a list of all the alerts available. Note that it is not essential that each system being monitored have the alerts match their actual hardware and software configuration. If an alert has no input the alert will be marked as invalid. Consider, for example, a disk mirroring alert. If the host state does not show that any disk mirroring exists on the host, then the disk mirroring alert would be invalid and ignored by the system. Thus, alerts that reference elements or token parameters not found in a particular host state are marked as invalid and ignored.

Note that the design of the alert system is intended to mirror a support engineers thought process. That is, when presented a problem, a number of system conditions would be checked for existence or correctness, a weighted judgment would be given after each investigation, eventually the final prognosis would be given.

In addition to generating the alerts, the existence of the alerts is communicated to, e.g., a support engineer. Referring to FIG. **2**, several features are provided to support the engineer responsible for a particular monitored system. For instance, in order to provide the information to a support engineer, one embodiment of the invention utilizes a JAVA Graphical Users Interface (GUI) application to display the alerts in alert display **245**. In this embodiment the GUI provides the support engineer with a number options for displaying alerts. For example, the GUI can, in one embodiment, display a list of all alerts that have arisen and have not been dealt with. The GUI could also provide the capability to perform various operations on a list of alerts, such as to filter the list by priority, customer and type of alert. The GUI could also allow the engineer to focus on certain customers, ignoring others. It will use personal configurations for the engineer that have been created through the configuration editor to access this functionality.

A configuration editor **227** stores engineer specific information about the system visualizer and the alert viewer. The configuration editor allows configuration of various aspects, such as which other remote monitoring sites (e.g., in other countries) the visualizer and alert viewer are to communicate with, as well as which monitored computer systems the engineer is responsible for. The configuration editor will also allow the engineer to define which applications start up by default.

The alert viewer can thus provide a scrolling list of alerts for customers specified by the local configuration file. The alert viewer displays such information as alert priority, customer name, alert type, host machine; time passed since alert raised. Color may also be used to distinguish varying levels of alert importance.

The support engineer also has a background task operating, the expert watch **241**, which in a UNIX embodiment is a daemon process that runs on the engineer's machine. Expert watch **241** monitors incoming alerts generated in alert analyzer **223** and when the expert watch **241** matches an alert type and customer with the engineer's own configuration profile, it will notify the engineer and cause the system visualizer to display the problem system at the point in the hierarchy where the problem exists. The problem would be shown graphically. If the system visualizer was not running, the expert watch daemon could case the system visualizer to start.

Alerts can be generated in another fashion other than the alert analyzer **223**, specifically phone home processing. Phone home processing is when a serious problem occurs on a monitored system requiring immediate attention, and the monitored system immediately contacts the service center via dial up modem or email and the like. Phone home processing **249** converts the incoming phone home messages into alerts. The alerts are then dealt as high priority alerts through the system. The alerts can be viewed by the alert viewer and/or emails are sent to the appropriate email addresses

In addition to notifying service engineers by displaying alerts, the alert processing in **247** may also generate email. A database such as the profile database **107** shown in FIG. **1** may include email addresses associated with particular monitored systems. When an alert of a predetermined seriousness occurs, an email is sent to the appropriate email addresses.

The description of the invention set forth herein is illustrative, and is not intended to limit the scope of the invention as set forth in the following claims. For instance, while exemplary embodiments were described in terms computers operating in a UNIX environment, the invention is also applicable to various computers utilizing other operating system and any time of processors and software. In light of the full scope of equivalence of the following claims, variations and modifications of the embodiments disclosed herein, may be made based on the description set forth herein, without departing from the scope and spirit of the invention as set forth in the following claims.

TABLE 3

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Host	Hostname		showrev-a	showrev-a			showrev-a	showrev-a	
	Platform	uname-a	uname-a	uname-a		uname-a	uname-a	uname-a	
	Host Id		showrev-a	showrev-a			showrev-a	showrev-a	
	Serial Number	ReMon Extract	ReMon Extract	configd		ReMon Extract	ReMon Extract	configd	
	Data Date	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract	ReMon Extract
	OS Release		showrev-a	showrev-a			showrev-a	showrev-a	
	Kernel Architecture		showrev-a	showrev-a			showrev-a	showrev-a	
	Application Architecture		showrev-a	showrev-a			showrev-a	showrev-a	
	Hardware Provider		showrev-a	showrev-a			showrev-a	showrev-a	
	Network Domain		showrev-a	showrev-a			showrev-a	showrev-a	
	Kernel Version		showrev-a	showrev-a			showrev-a	showrev-a	
	Openwindows Version		showrev-a	showrev-a			showrev-a	showrev-a	
	System Clock Frequency		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
Hardware Information	Hardware Information (D)	D	D	D	D	D	D	D	D
	Application Architecture		showrev-a	showrev-a			showrev-a	showrev-a	
System Board	System Board ID		vtsprobe	vtsprobe		D	D	D	
	type	ND	ND	configd		ND	ND	configd	
	State	ND	ND	configd		ND	ND	configd	
	slot number	ND	ND	configd		ND	ND	ND	
	temperature	ND	ND	configd		ND	ND	ND	
	board reference number	ND	ND	configd		ND	ND	configd	
Memory Controller	Memory Controller ID	D	D	configd		D	D	D	
	memory controller model	ND	ND	configd		ND	ND	ND	
Memory	Memory Value		vtsprobe	configd			vtsprobe	configd	
Memory Simm	Memory Simm ID	ND	ND	configd		ND	ND	configd	
	board reference number	ND	ND	configd		ND	ND	configd	
	size	ND	ND	configd		ND	ND	configd	
CPU	CPU Unit ID		vtsprobe	configd			vtsprobe	configd	
	Clock frequency		vtsprobe	configd			vtsprobe	configd	
	CPU model		vtsprobe	configd			vtsprobe	configd	
	CPU dcache size	ND	ND	configd		ND	ND	configd	
	CPU ecache size	ND	ND	configd		ND	ND	configd	
	CPU icache size	ND	ND	configd		ND	ND	configd	
	CPU status	ND	ND	configd		ND	ND	configd	
EEPROM	EEPROM Model	D	D	configd		D	D	D	
	FlashProm model	ND	ND	configd		ND	ND	ND	
Peripheral Bus	peripheral bus id		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	Model No	ND	ND	configd		ND	ND	configd	
	Registration Number	ND	ND	configd		ND	ND	configd	
Audio Port	Audio Port Id		vtsprobe	configd				configd	
Audio Port Error	Audio Port Error String		audio-lv	audio-lv			audio-lv	audio-lv	
Parallel Port	parallel port id		vtsprobe	configd			vtsprobe	configd	
ParallelPortError	Parallel Port Error String		bpptest-lv	bpptest-lv			bpptest-lv	bpptest-lv	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Serial Port	serial port id		vtsprobe	configd			vtsprobe	configd	
Diskette	Diskette ID		vtsprobe	configd			vtsprobe	configd	
	Diskette Status	ND	ND	configd		ND	ND	configd	
	Diskette Type	ND	ND	configd		ND	ND	configd	
Peripheral Adaptor	Peripheral Adaptor ID		vtsprobe	configd			vtsprobe	configd	
	peripheral adaptor model name		vtsprobe	configd			vtsprobe	configd	
	peripheral adaptor type		vtsprobe	configd			vtsprobe	configd	
	sbus slot no	ND	ND	configd		ND	ND	configd	
	speed register	ND	ND	configd		ND	ND	configd	
CDROM	CDROM ID		vtsprobe	configd			vtsprobe	configd	
Tape	TAPE ID		vtsprobe	configd			vtsprobe	configd	
	Tape Type		vtsprobe	configd			vtsprobe	configd	
Tape Hardware Errors	Tape HW Error String		tapetest-lv	tapetest=lv			tapetest-lv	tapetest-lv	
SCSI Disk	SCSI Disk ID		vtsprobe	configd			vtsprobe	configd	
	SCSI Disk Sectors per track		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI disk firmware rev		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI disk serial number		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Sectors per Cylinder		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Sun ID		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Cylinders		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Capacity		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Software Controller		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Accessible Cylinders		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Tracks per Cylinder		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Bytes per Sector		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	SCSI Disk Vendor		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
SCSI Disk Error	DIsk Error String		disktest-lv (o)	disktest-lv (o)			disktest-lv (o)	disktest-lv (o)	
SCSI Disk Partition	SCSI DIsk Partition ID	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
	SCSI DIsk Partition last sector	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
	SCSI Disk Partition Sector Count	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
	Scsi Disk Partition First Sector	diskprobe	diskprobe	diskprobe		diskprobe	diskprobe	diskprobe	
SCSI DIsk Bad Block	SCSI BAD Block ID	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg
	Time occurred	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg	dmesg
Network Adaptor	Network Adapator ID		vtsprobe	configd			vtsprobe	configd	
	Internet Address		vtsprobe	configd			vtsprobe	configd	
	sbus slot no	ND	ND	configd		ND	ND	configd	
Network Hardware Error	Network HW Error String		nettest-lv (o)	nettest-lv (o)			nettest-lv (o)	nettest-lv (o)	
Serial Optical Channel Processor Host Adaptor	Serial Optical Processor ID	ND	ND	configd		ND	ND	configd	
	Sbus slot number	ND	ND	configd		ND	ND	configd	
	SO model No.	ND	ND	configd		ND	ND	configd	
Storage Array									
Storage Array Disk									
Storage Array Partition									

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Graphics Adaptor	Graphics Adaptor ID		vtsprobe	configd			vtsprobe	configd	
	graphics adaptor model no	ND	ND	configd		ND	ND	configd	
	sbus slot number	ND	ND	configd		ND	ND	configd	
Monitor	Monitor Dtype (D)	D	D	D	D	D	D	D	D
Serial Port Expander									
System Board PSU	Power Supply ID	ND	ND	configd		ND	ND	ND	
	Power Supply Wattage	ND	ND	configd		ND	ND	ND	
	Power Supply Status	ND	ND	configd		ND	ND	ND	
System Power Supply	System Power Supply (D)	D	D	D		D	D	D	
Software packages & patches	Software Packages & packages (D)	D	D	D	D	D	D	D	D
Software Package	Software Package ID		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Vendor		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Files		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Category		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Architecture		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Status		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Base Directory		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Version		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package pstamp		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Installation data		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
	Package Name		pkginfo-l	pkginfo-l			pkginfo-l	pkginfo-l	
Software Patch	Software Patch ID	showrev-p	showrev-p	showrev-p		showrev-p	showrev-p	showrev-p	
	Revision ID	showrev-p	showrev-p	showrev-p		showrev-p	showrev-p	showrev-p	
Operating System	Operating System ID	showrev-a	showrev-a	showrev-a		showrev-a	showrev-a	showrev-a	
	OS Release Type	showrev-a	showrev-a	showrev-a		showrev-a	showrev-a	showrev-a	
	OS Release Version	showrev-a	showrev-a	showrev-a		showrev-a	showrev-a	showrev-a	
System Services	System Services (D)	D	D	D	D	D	D	D	D
Unix Filesystems	Unix Filesystems (D)	D	D	D	D	D	D	D	D
Local Filesystems (Boot)	Local Filesystem name	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab
	Associative Device Name	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab
	mount point	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab
	FSCK pass	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab
	mount options	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab	more/etc/ vfstab
Local Filesystem (Current)	Local Current Filesystem Name	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Associative Device Name	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	mount point	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	mount options	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Date mounted since	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Total Kbytes available	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk
	Total Kbytes used	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk
	Percentage Capacity	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk	df-lk

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Files used	df-l	df-l	df-i	df-l	df-l	df-l	df-l	df-l
Cache FS	Cachefs Daerrion	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Cached Filesystem (Boot)	Cached Filesystem Id	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	Filesystem cached	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	mount options	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
Cached File System (Current)	Cached Filesystem ID	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Filesystem cached	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	cache bits	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat
	cache misses	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat
	cache modifies	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat	cachefsstat
NFS Server	NFS Server exists	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Exported Filesystem (boot)	Exported Filesystem name	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab
	Export options	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab
	Export Description	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab	more/etc/dfs/dfstab
Exported Filesystem (current)	Exported filesystem name	share	share	share	share	share	share	share	share
	Exported file system options	share	share	share	share	share	share	share	share
	Exported FS description	share	share	share	share	share	share	share	share
NFS Client	Filesystems being mounted	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Mounted Filesystem (Boot)	Filesystem Name	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	remote machine	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	remote filesystem	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	Mount type (kerberos/NFS)	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	mount point	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
	mount permissions	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab	more/etc/vfstab
Mounted Filesystem (Current)	Filesystem Name	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Remote machine	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	remote filesystem	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	mount type	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	mount point	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Date mounted since	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v	mount-v
	Total Kbytes available	df-k	df-k	df-k	df-k	df-k	df-k	df-k	df-k
	Total Kbytes used	df-k	df-k	df-k	df-k	df-k	df-k	df-k	df-k
	Percentage Capacity	df-k	df-k	df-k	df-k	df-k	df-k	df-k	df-k
	Files used	df	df	df	df	df	df	df	df

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
NIS Server	NIS Server in existance	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	domainname	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a
NIS client	NIS client software exists	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	domainname	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a
	Server Bound to	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich	ypwhich
NIS+Master	configured as NIS+Master	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	domainname	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	access rights	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
NIS+Replica	configured as a NIS+Replica	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	domainname	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	access rights	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	NIS+Master	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
NIS+Client	configured as a NIS+client	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	domainname	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
	NIS+Master	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
NIS+Search Path	NIS+Search Path	niscat-o org_dir	niscat-o org_dir	niscat-o org_dir		niscat-o org_dir	niscat-o org_dir	niscat-o org_dir	
DNS client	DNS being used	ps-ef	ps-ef	ps-ef		ps-ef	ps-ef	ps-ef	
DNS resolve hosts	DNS resolve host id	more/etc/ resolv.conf	more/etc/ resolv.conf	more/etc/ resolv.conf		more/etc/ resolv.conf	more/etc/ resolv.conf	more/etc/ resolv.conf	
NameService Configuration	String Dummy Token	D	D	D	D	D	D	D	D
Password Map Resolve Type	Name Service map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
Group Map Resolve Type	Name Service map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
Hosts Map Resolve Type	Name Service map resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
Protocols Map Resolve Type	Name Service map resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
RPC Map Resolve Type	Name Service map resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
Ethers Map Resolve Type	Name Service map resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
Netmasks Map Resolve Type	Name Service map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
boorparams Map Resolve Type	Name Service map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
publickey Map Resolve Type	Name Service map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
netgroup Map Resolve Type	Name Service Map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
automount Map resolve Type	Name Service Map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
aliases Map resolve Type	Name Service Map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
services Map resolve Type	Name Service Map Resolved by	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
sebdmailvars resolve Type	Name Service Map Resolved By	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf		more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	more/etc/ nss- witch.conf	
CRON	cron is running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
root cronjob	cronjob name	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1
	cronjob time control-string	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1
	cronjob execution string	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1	crontab-1
Sendmail	sendmail is running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	major relay mailer	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf
	major relay host (DR)	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf
	major relay host (CR)	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf	more/etc/ mail/send- mai.cf
TCP/IP	Name of Ethernet board		vtsprobe	configd			vtsprobe	configd	
	Internet Address	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a
Routing Table	Dummy Token	D	D	D	D	D	D	D	D
Network Route	Network Route Destination	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r
	Gateway	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		≥ Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Flags	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r
	use	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r	netstat-r
Network Known host list	Dummy Token	D	D	D	D	D	D	D	D
Network Known host	Known IP address	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a
	Network Mask	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a
	Physical Address	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a	arp-a
Javastation boot list									
Javastation									
Volume manager	Create if running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Kernel	Kernel Architecture	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a	showrev-a
	Kernel memory		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
Kernel Memory Leak	Value leaked		kmemleak	kmemleak			kmemleak	kmemleak	
Kernel module list	Dummy Token	D	D	D	D	D	D	D	D
kernel module	Kernel Module Name	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
	Kernel Module Load address	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
	Kernel Module Info	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
	Kernel Module Revision	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo	modinfo
kernel statistics	dummy Token	D	D	D	D	D	D	D	D
VMHAT	dummy token	D	D	D	D	D	D	D	D
	VH CTX Free		netstat-k	netstat-k			netstat-k	netstat-k	
	VH CTX Dirty		netstat-k	netstat-k			netstat-k	netstat-k	
	VH CTX steal		netstat-k	netstat-k			netstat-k	netstat-k	
	VH TTE load		netstat-k	netstat-k			netstat-k	netstat-k	
	VH page faults		netstat-k	netstat-k			netstat-k	netstat-k	
	VH steal count		netstat-k	netstat-k			netstat-k	netstat-k	
Segment map	Dummy text	D	D	D	D	D	D	D	D
	Faults		netstat-k	netstat-k			netstat-k	netstat-k	
	Faults		netstat-k	netstat-k			netstat-k	netstat-k	
	getmap		netstat-k	netstat-k			netstat-k	netstat-k	
	Page Create		netstat-k	netstat-k			netstat-k	netstat-k	
Buffer IO	Dummy Token	D	D	D	D	D	D	D	D
	Buffer Cache Lookups		netstat-k	netstat-k			netstat-k	netstat-k	
	Buffer Cache bits		netstat-k	netstat-k			netstat-k	netstat-k	
	Waits for Buffer Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
	Duplicate Buffers found		netstat-k	netstat-k			netstat-k	netstat-k	
System pages	Dummy Token	D	D	D	D	D	D	D	D
	Physical Memory		netstat-k	netstat-k			netstat-k	netstat-k	
	Inalloc		netstat-k	netstat-k			netstat-k	netstat-k	
	nFree		netstat-k	netstat-k			netstat-k	netstat-k	
	kernel base		netstat-k	netstat-k			netstat-k	netstat-k	
	econtig		netstat-k	netstat-k			netstat-k	netstat-k	
	free memory		netstat-k	netstat-k			netstat-k	netstat-k	
	available rmem		netstat-k	netstat-k			netstat-k	netstat-k	
	pages free		netstat-k	netstat-k			netstat-k	netstat-k	
	pages locked		netstat-k	netstat-k			netstat-k	netstat-k	
RPC CLTS Client	Dummy Token	D	D	D	D	D	D	D	D

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Calls		netstat-k	netstat-k			netstat-k	netstat-k	
	Badcalls		netstat-k	netstat-k			netstat-k	netstat-k	
	badxids		netstat-k	netstat-k			netstat-k	netstat-k	
	timeouts		netstat-k	netstat-k			netstat-k	netstat-k	
RPC COTS Client	Dummy Token	D	D	D	D	D	D	D	D
	Calls		netstat-k	netstat-k			netstat-k	netstat-k	
	badcalls		netstat-k	netstat-k			netstat-k	netstat-k	
	badxids		netstat-k	netstat-k			netstat-k	netstat-k	
	interrupts		netstat-k	netstat-k			netstat-k	netstat-k	
RPC COTS Connections	Dummy Token	D	D	D	D	D	D	D	D
	Write Queue		netstat-k	netstat-k			netstat-k	netstat-k	
	Server		netstat-k	netstat-k			netstat-k	netstat-k	
	status		netstat-k	netstat-k			netstat-k	netstat-k	
RPC Client	Dummy Token	D	D	D	D	D	D	D	D
	calls		netstat-k	netstat-k			netstat-k	netstat-k	
	badcalls		netstat-k	netstat-k			netstat-k	netstat-k	
	re transmits		netstat-k	netstat-k			netstat-k	netstat-k	
	badxids		netstat-k	netstat-k			netstat-k	netstat-k	
	can't send		netstat-k	netstat-k			netstat-k	netstat-k	
RPC CLTS Server	Dummy Token	D	D	D	D	D	D	D	D
	Calls		netstat-k	netstat-k			netstat-k	netstat-k	
	badcalls		netstat-k	netstat-k			netstat-k	netstat-k	
	xdr call		netstat-k	netstat-k			netstat-k	netstat-k	
RPC COTS Server	Dummy Token	D	D	D	D	D	D	D	D
	calls		netstat-k	netstat-k			netstat-k	netstat-k	
	badcalls		netstat-k	netstat-k			netstat-k	netstat-k	
	xdr call		netstat-k	netstat-k			netstat-k	netstat-k	
RPC Server	Dummy Token	D	D	D	D	D	D	D	D
	calls		netstat-k	netstat-k			netstat-k	netstat-k	
	bad calls		netstat-k	netstat-k			netstat-k	netstat-k	
	xdr calls		netstat-k	netstat-k			netstat-k	netstat-k	
Inode cache	Dummy Token	D	D	D	D	D	D	D	D
	size		netstat-k	netstat-k			netstat-k	netstat-k	
	maxsize		netstat-k	netstat-k			netstat-k	netstat-k	
	hits		netstat-k	netstat-k			netstat-k	netstat-k	
	misses		netstat-k	netstat-k			netstat-k	netstat-k	
	mallocs		netstat-k	netstat-k			netstat-k	netstat-k	
Kernel Mmory magazine	Magazine ID		netstat-k	netstat-k			netstat-k	netstat-k	
	Buffer Size		netstat-k	netstat-k			netstat-k	netstat-k	
	buffer available		netstat-k	netstat-k			netstat-k	netstat-k	
	alloc fail		netstat-k	netstat-k			netstat-k	netstat-k	
kernel memory buffer control cache	Dummy Token	D	D	D	D	D	D	D	D
		Buffer Size		netstat-k	netstat-k			netstat-k	netstat-k
	buffer available		netstat-k	netstat-k			netstat-k	netstat-k	
kernel memory allocation	Kernel memory allocation ID		netstat-k	netstat-k			netstat-k	netstat-k	
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	buffer size		netstat-k	netstat-k			netstat-k	netstat-k	
	buffer total		netstat-k	netstat-k			netstat-k	netstat-k	
SFMMU Cache	SFMMU Cache ID		netstat-k	netstat-k			netstat-k	netstat-k	
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Failed Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
Segment Cache	Dummy Token		netstat-k	netstat-k			netstat-k	netstat-k	
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Failed Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
Thread Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat-k	netstat-k			netstat-k	netstat-k	
	Failed Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
Leight Weight Process Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat-k	netstat-k			netstat-k	netstat-k	
	Failed Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
CRED Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available	netstat-k	netstat-k			netstat-k	netstat-k		
	Failed Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
Thread Cache	Dummy Token	D	D	D	D	D	D	D	D
	buffer available		netstat-k	netstat-k			netstat-k	netstat-k	
	Failed Allocations		netstat-k	netstat-k			netstat-k	netstat-k	
	global allocations		netstat-k	netstat-k			netstat-k	netstat-k	
Steams Message	Steam Message ID		netstat-k	netstat-k			netstat-k	netstat-k	
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
Stream Head Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
Queue Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
Sync Q Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
Streams Message	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
anonymous Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
anonymous map Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
seg VN Cache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
UFS InodeCache	Dummy Data	D	D	D	D	D	D	D	D

TABLE 3-continued

Element Entries		Server				Desktop			
		Legacy		Enterprise		Legacy		>= Ultra 2	
Element	Token Type	5.4	5.5	5.5.1	5.6	5.4	5.5	5.5.1	5.6
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
FASCache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
PipeCache	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
LM sysid	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
LM client	Dummy Data	D	D	D	D	D	D	D	D
	Buffer Available		netstat-k	netstat-k			netstat-k	netstat-k	
	Allocation Failures		netstat-k	netstat-k			netstat-k	netstat-k	
Virtual Memory	Total Virtual Memory size		vtsprobe	vtsprobe			vtsprobe	vtsprobe	
	Virtual Memory Free	vmstat	vmstat	vmstat		vmstat	vmstat	vmstat	
Page Fault	Page fault id	vmstat	vmstat	vmstat		vmstat	vmstat	vmstat	
Windowing System	Windowing system revision	showrev-a	showrev-a	showrev-a		showrev-a	showrev-a	showrev-a	
	Display Size	xdpyinfo	zdpyinfo	zdpyinfo		zdpyinfo	zdpyinfo	xdpyinfo	
	Depth of root window	xdpyinfo	xdpyinfo	xdpyinfo		xdpyinfo	xdpyinfo	xdpyinfo	
	resolution	xdpyinfo	xdpyinfo	xdpyinfo		xdpyinfo	xdpyinfo	xdpyinfo	
Process Table	Dummy Token	D	D	D	D	D	D	D	D
Process	process name	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	time	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	process id	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	parent id	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Unbundled Software	Dummy Token	D	D	D	D	D	D	D	D
Solstice DiskSuite									
MetaDisk Partition	Solstice Backup								
Solstice Backup partition									
Solstice Symon	installed	ND	ND	pkginfo-1	pkginfo-1	ND	ND	pkginfo-1	pkginfo-1
	Symond running	ND	ND	ps-ef	ps-ef	ND	ND	ps-ef	ps-ef
	Kernel reader running	ND	ND	ps-ef	ps-ef	ND	ND	ps-ef	ps-ef
	Log scanner running	ND	ND	ps-ef	ps-ef	ND	ND	ps-ef	ps-ef
Sybase	Sybase datasrver running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
	sybase backup dataserver running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Oracle	Oracel server running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
Informix	Informix server running	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef	ps-ef
SAP									

Key:
(D) — Dummy Token
(ND) — No data available
(M) — Output of test must be modified as it is different across OS releases
*Blank spaces means untested or unknown tests.

What is claimed is:

1. A method comprising:

providing a host state representing a state of a computer system, the host state being represented as a modifiable tree structure including elements in a fixed hierarchical relationship, the elements being given value by associated tokens, the elements and associated tokens representing hardware and software components of the computer system and wherein the tokens are extracted from diagnostic data from the computer system;

determining if predetermined conditions exist in the computer system by comparing respective definitions of the predetermined conditions to the host state; and

generating an alert if one of the predetermined conditions is determined to exist.

2. The method as recited in claim 1 wherein each of the definitions is an alert definition defining a respective state of aspects of the computer system hardware and software.

3. The method as recited in claim 2 wherein at least one of the alert definitions generates an alert according to at least one token value.

4. The method as recited in claim 2 wherein at least one of the alert definitions generates an alert according to a value of at least one other alert.

5. The method as recited in claim 1 wherein the computer system is remote from a monitoring computer system on which the host state is provided.

6. The method as recited in claim 1 further comprising:

providing a plurality of host states representing the computer system, each of the host states representing the state of the computer system over a different time period;

extracting at least one token value from each of a number of said host states; and

comparing the extracted token values from said number of host states to the conditions defined in the definitions, thereby monitoring conditions existing in the computer system over time.

7. A method comprising:

providing a modifiable static tree structure representing a general computer system;

extracting component information indicating hardware and software components of the computer system, from diagnostic data of the computer system;

generating a host state, representing a state of a computer system, according to the static tree structure and the component information, wherein the static tree structure includes element types in a fixed hierarchical relationship, the element types representing the hardware and software components of the computer system;

determining if predetermined conditions exist in the computer system by comparing respective definitions of the predetermined conditions to the host state; and

generating an alert if one of the predetermined conditions is determined to exist.

8. The method as recited in claim 7 further comprising extracting the component information as a plurality of tokens, the tokens being respective instances of respective token types, each of the token types having a value of one aspect of the component information and an indication of an association with one of the elements in the static tree.

9. The method as recited in claim 7 wherein the computer system is part of a first computer system and the diagnostic data is communicated from the first computer system to a second computer system, the second computer system being remote from the first computer system, the second computer rebuilding the state of the computer system.

10. A monitoring computer system for generating alerts indicating predetermined conditions exist in a monitored computer system, comprising:

a first data storage area storing a plurality of alert definitions defining respective predetermined conditions in the monitored computer system;

a second data storage area storing at least a first host state of the monitored computer system, the first host state being represented as a modifiable tree structure including elements in a fixed hierarchical relationship, the elements being given value by associated token values indicating respective software and hardware components of the monitored computer system; and

a monitoring computer, coupled to the first and second data storage areas; and

wherein the monitoring computer generates alerts when a condition defined in one of the alert definitions is determined to be present in the first host state.

11. The monitoring computer as recited in claim 10 wherein the first host state is represented as a tree structure including elements in a fixed hierarchical relationship, the elements being given value by associated tokens, the elements and associated tokens representing hardware and software components of the computer system and wherein the tokens are extracted from diagnostic data provided from the monitored computer system.

12. The monitoring computer as recited in claim 10 wherein the monitored computer system is remotely located from the monitoring computer system and wherein the diagnostic data is provided from the remotely located monitored computer system to the monitoring computer system.

13. The monitoring computer system as recited in claim 11 further comprising a third data storage area coupled to the monitoring computer, the third data storage area storing a plurality of host states representing the monitored computer system, the plurality of host states in addition to the first host state, each of the first and plurality of host states representing the state of the monitored computer system over a different period of time.

14. The monitoring computer system as recited in claim 13 wherein at least one of the alert definitions defines an alert in terms of a value of tokens over time, the value of tokens over time being obtained from the first host state and the plurality of host states.

15. The monitoring computer system as recited in claim 13 wherein at least one of the alert definitions defines a spot alert in terms data stored only in the first host state, the first host state being the latest in time.

16. A computer program stored in computer readable media and operable on a monitoring computer system to evaluate the state of a monitored computer system, the computer program:

comparing a plurality of alert definitions, defining predetermined conditions existing on a computer system, to at least one host state representing a state of a computer system, wherein the host state is represented as a tree structure including elements in a modifiable hierarchical relationship, the elements being given value by

associated tokens, the elements and associated tokens representing hardware and software components of the computer system and wherein the tokens are extracted from diagnostic data from the computer system;

generating an alert if the conditions defined in one of the alert definitions exists in the host state.

17. The computer program as recited in claim **16** wherein each generated alert provides an associated severity level, thereby indicating seriousness of a problem detected by the respective alert.

18. The computer program as recited in claim **16** further including an alert definition generating an alert according to data contained in the latest host state, the data being contained in at least one token.

19. The computer program as recited in claim **16** further comprising a predictive alert definition, the predictive alert definition defining a condition based on data contained in a plurality of host states, each of the host states representing the state of the computer system over a different time period.

20. The computer program as recited in claim **19** wherein the predictive alert is based upon a rate of change of the data contained in the plurality of host states.

* * * * *

US006484200B1

(12) **United States Patent**
Angal et al.

(10) **Patent No.: US 6,484,200 B1**
(45) **Date of Patent: Nov. 19, 2002**

(54) **DISTINGUISHED NAME SCOPING SYSTEM FOR EVENT FILTERING**

(75) Inventors: **Rajeev Angal**, Santa Clara, CA (US); **Shivaram Bhat**, Sunnyvale, CA (US); **Michael Roytman**, Glenview, IL (US); **Subodh Bapat**, Palo Alto, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/330,790**

(22) Filed: **Jun. 11, 1999**

(51) **Int. Cl.**[7] **H04L 12/26**
(52) **U.S. Cl.** **709/224**; 709/218; 370/352
(58) **Field of Search** 709/203, 224, 709/227, 232, 216, 218, 206; 370/352; 710/10, 4

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,537,611 A 7/1996 Rajagopal et al. 379/221.07
5,678,041 A * 10/1997 Baker et al. 709/227
5,848,243 A * 12/1998 Kulkarni et al. 709/224
6,141,777 A 10/2000 Cutrell et al.
6,147,975 A * 11/2000 Bowman-Amuah 370/352
6,182,119 B1 1/2001 Chu
6,249,883 B1 6/2001 Cassidy et al.

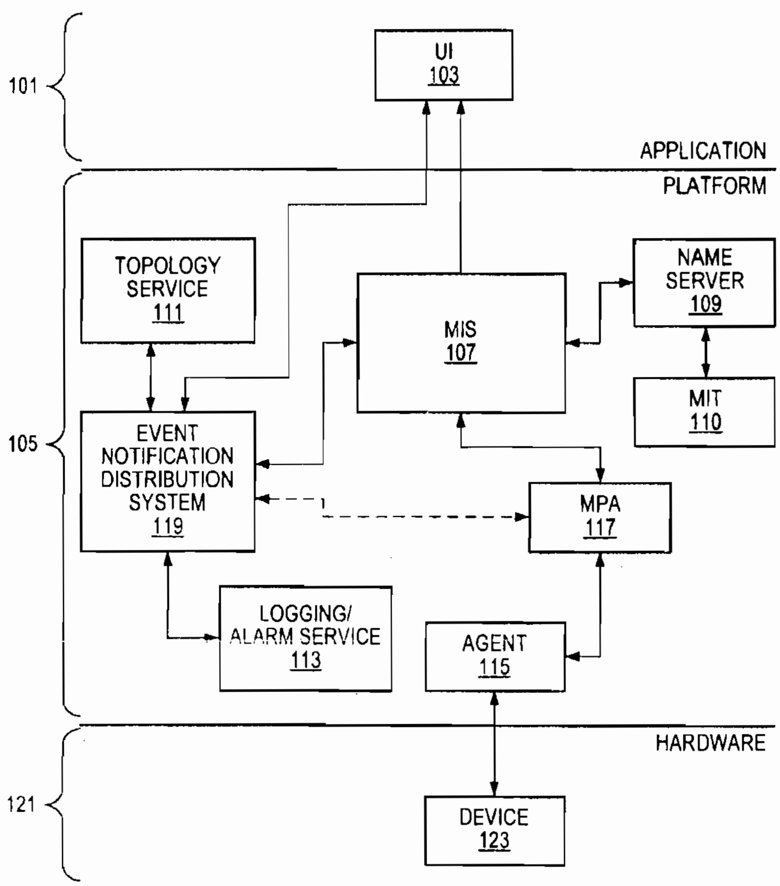
* cited by examiner

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Khanh Quang Dinh
(74) *Attorney, Agent, or Firm*—B. Noël Kivlin

(57) **ABSTRACT**

Method and system for allowing a computer network operations manager to subscribe for and receive notifications concerning network events from one or more objects or object levels, as defined by distinguished name scoping, and optionally having at least one event characteristic from a selected list. The selected list of characteristics may include: one or more levels of network objects involved in the event; one or more specified network nodes involved in the event; a specified geographical region in which said event occurs; a specified period of days within which the event occurs or is initiated; a specified time interval within which the event occurs or is initiated; a specified class of devices involved in the event; and an event of one or more specified event types.

15 Claims, 6 Drawing Sheets



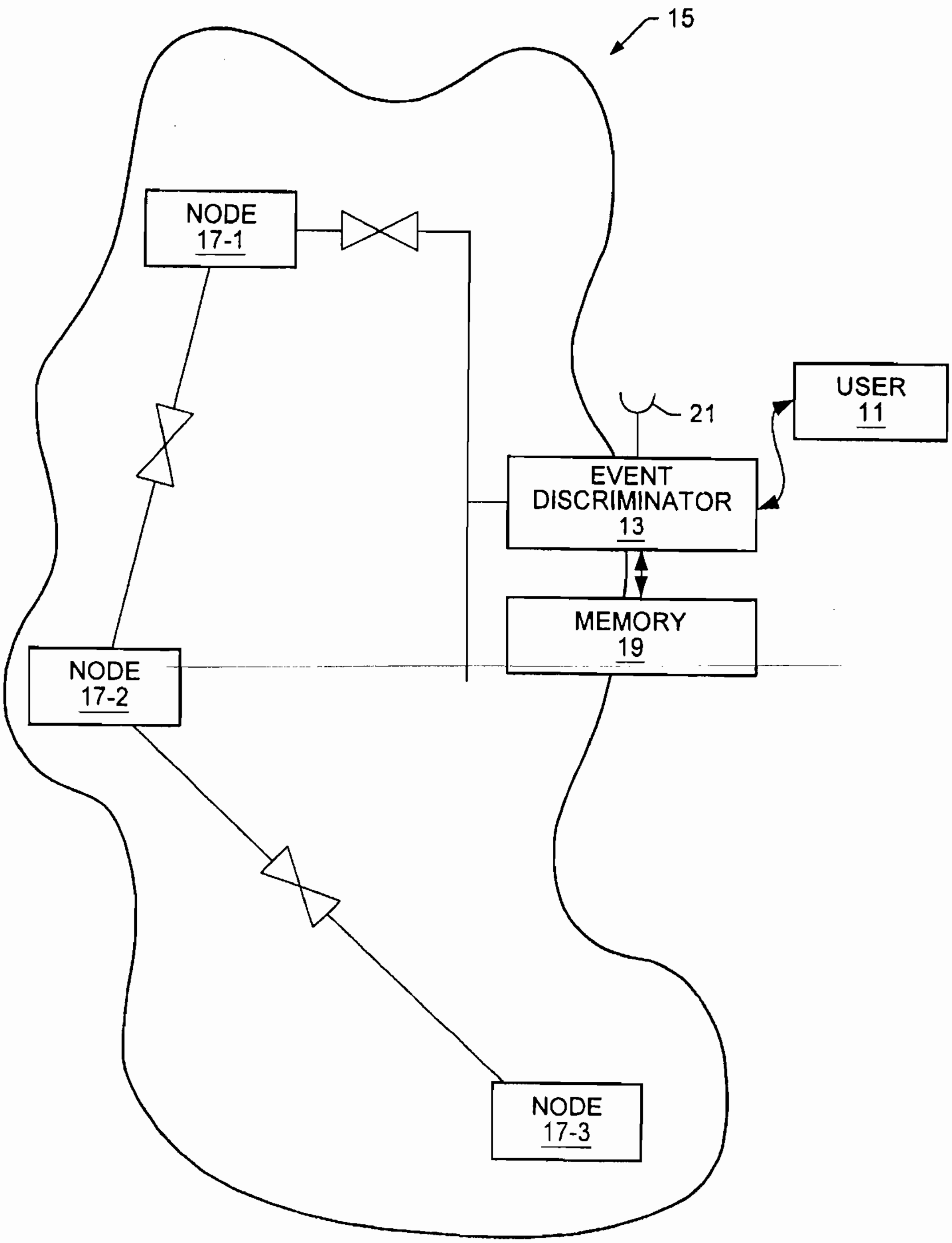

15
NODE 17-1
NODE 17-2
NODE 17-3
21
EVENT DISCRIMINATOR 13
MEMORY 19
USER 11

Fig. 1

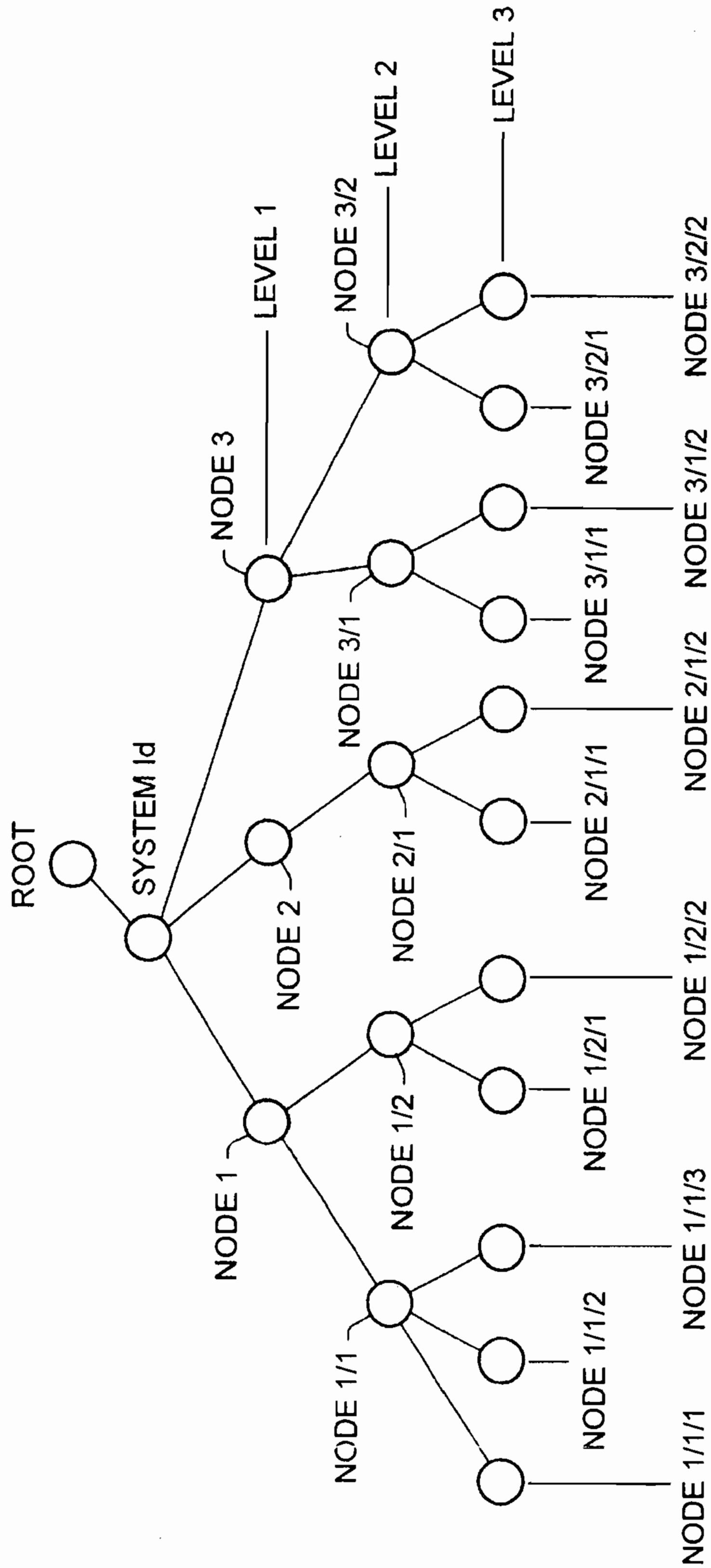

ROOT
SYSTEM Id
NODE 1
NODE 2
NODE 3
LEVEL 1
NODE 1/1
NODE 1/2
NODE 2/1
NODE 3/1
NODE 3/2
LEVEL 2
NODE 1/1/1
NODE 1/1/2
NODE 1/1/3
NODE 1/2/1
NODE 1/2/2
NODE 2/1/1
NODE 2/1/2
NODE 3/1/1
NODE 3/1/2
NODE 3/2/1
NODE 3/2/2
LEVEL 3

Fig. 2

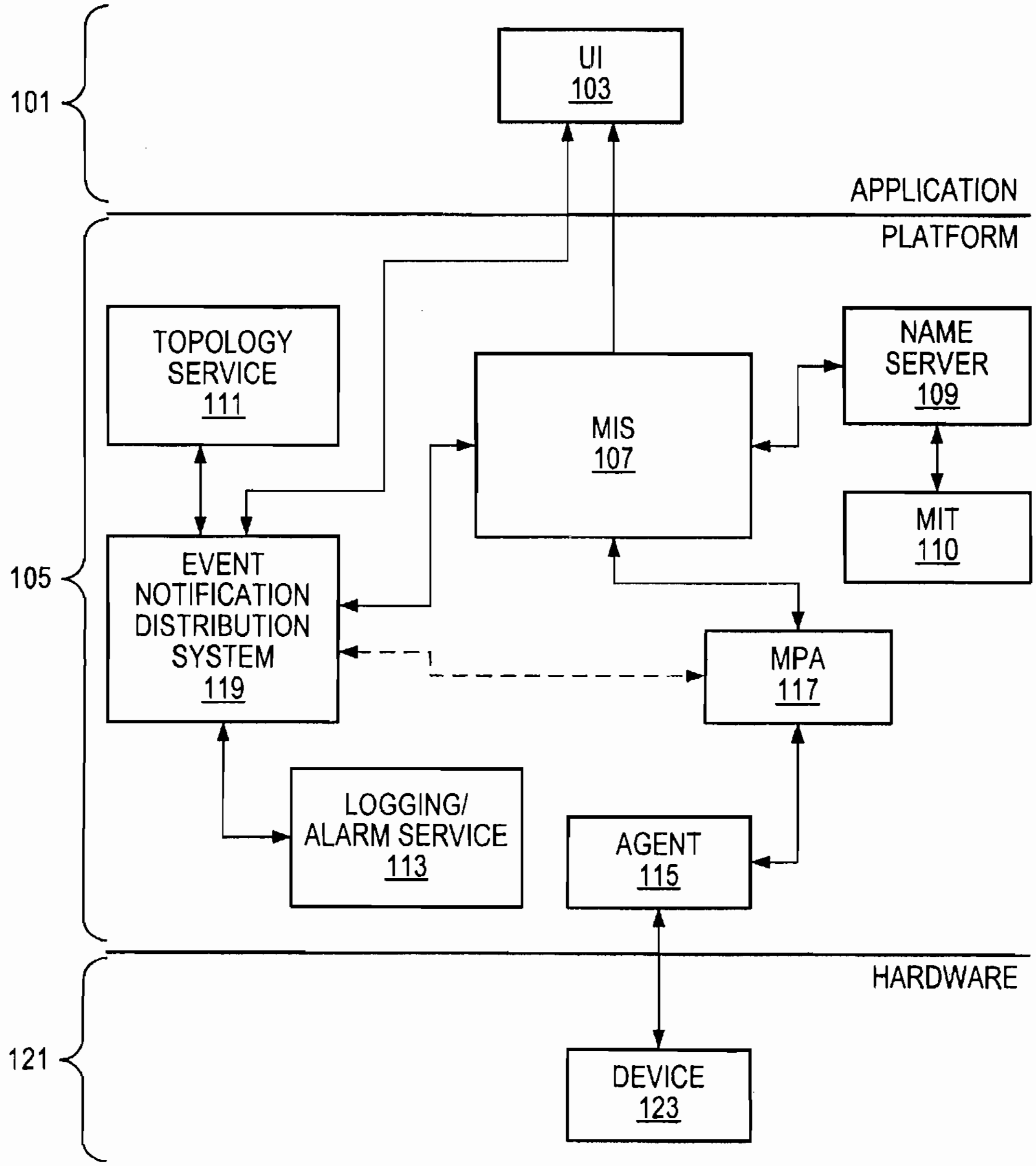

101
UI
103
APPLICATION
PLATFORM
TOPOLOGY SERVICE
111
MIS
107
NAME SERVER
109
MIT
110
105
EVENT NOTIFICATION DISTRIBUTION SYSTEM
119
MPA
117
LOGGING/ ALARM SERVICE
113
AGENT
115
HARDWARE
121
DEVICE
123

Fig. 3

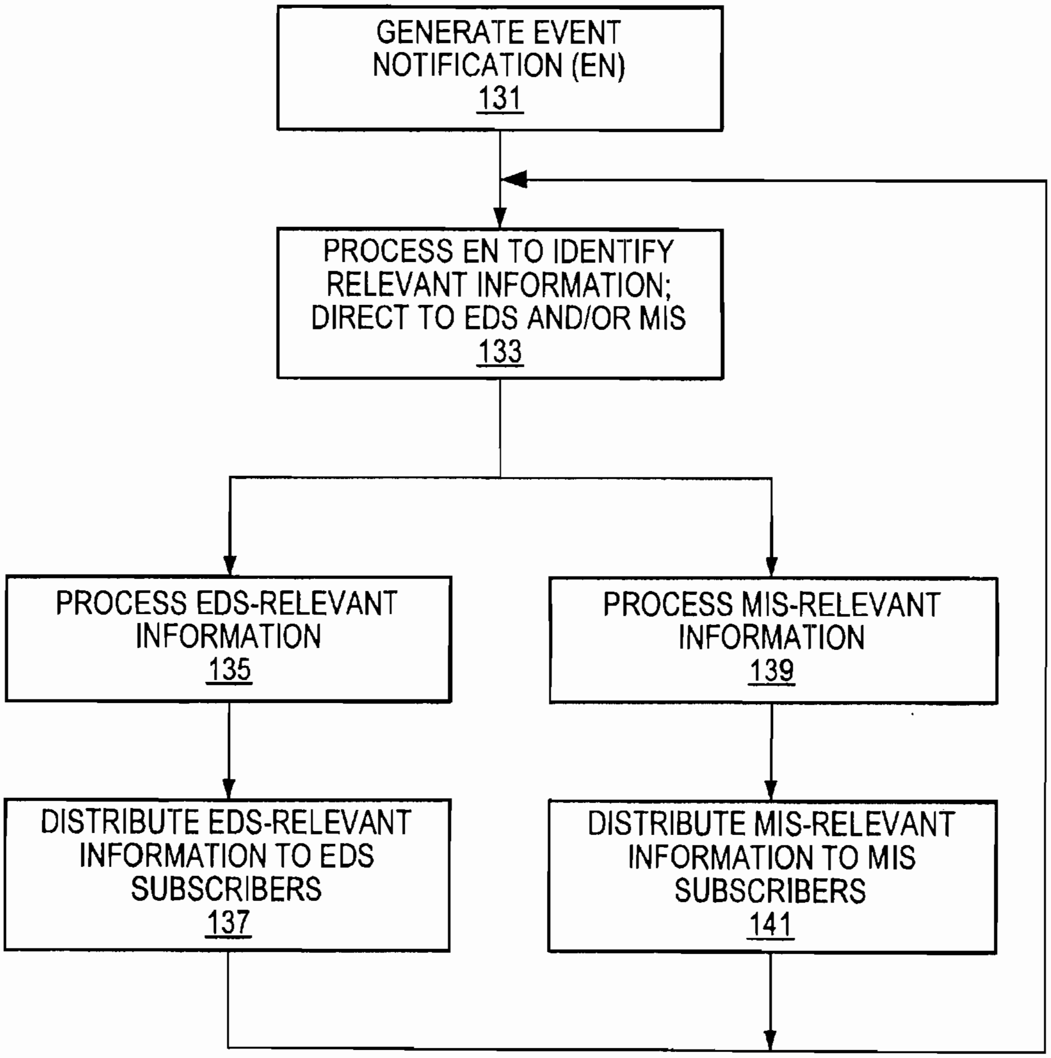

GENERATE EVENT NOTIFICATION (EN)
131
PROCESS EN TO IDENTIFY RELEVANT INFORMATION; DIRECT TO EDS AND/OR MIS
133
PROCESS EDS-RELEVANT INFORMATION
135
PROCESS MIS-RELEVANT INFORMATION
139
DISTRIBUTE EDS-RELEVANT INFORMATION TO EDS SUBSCRIBERS
137
DISTRIBUTE MIS-RELEVANT INFORMATION TO MIS SUBSCRIBERS
141

Fig. 4

EVENT CHARACTERISTICS SPECIFICATION

CHARACTERISTIC	PARAMETER(S)	DON'T CARE (x)
OBJECT LEVEL INDICATOR (SINGLE)		
OBJECT LEVEL INDICATORS (MULTIPLE)		
EVENT SOURCE (NODES)		
EVENT SOURCE (GEOGRAPHICAL REGION)		
DATA OF EVENT OCCURRENCE		
TIME INTERVAL OF EVENT OCCURRENCE		
DEVICE TYPE(S) INVOLVED		
EVENT SEVERITY		
EVENT TYPE		

Fig. 5

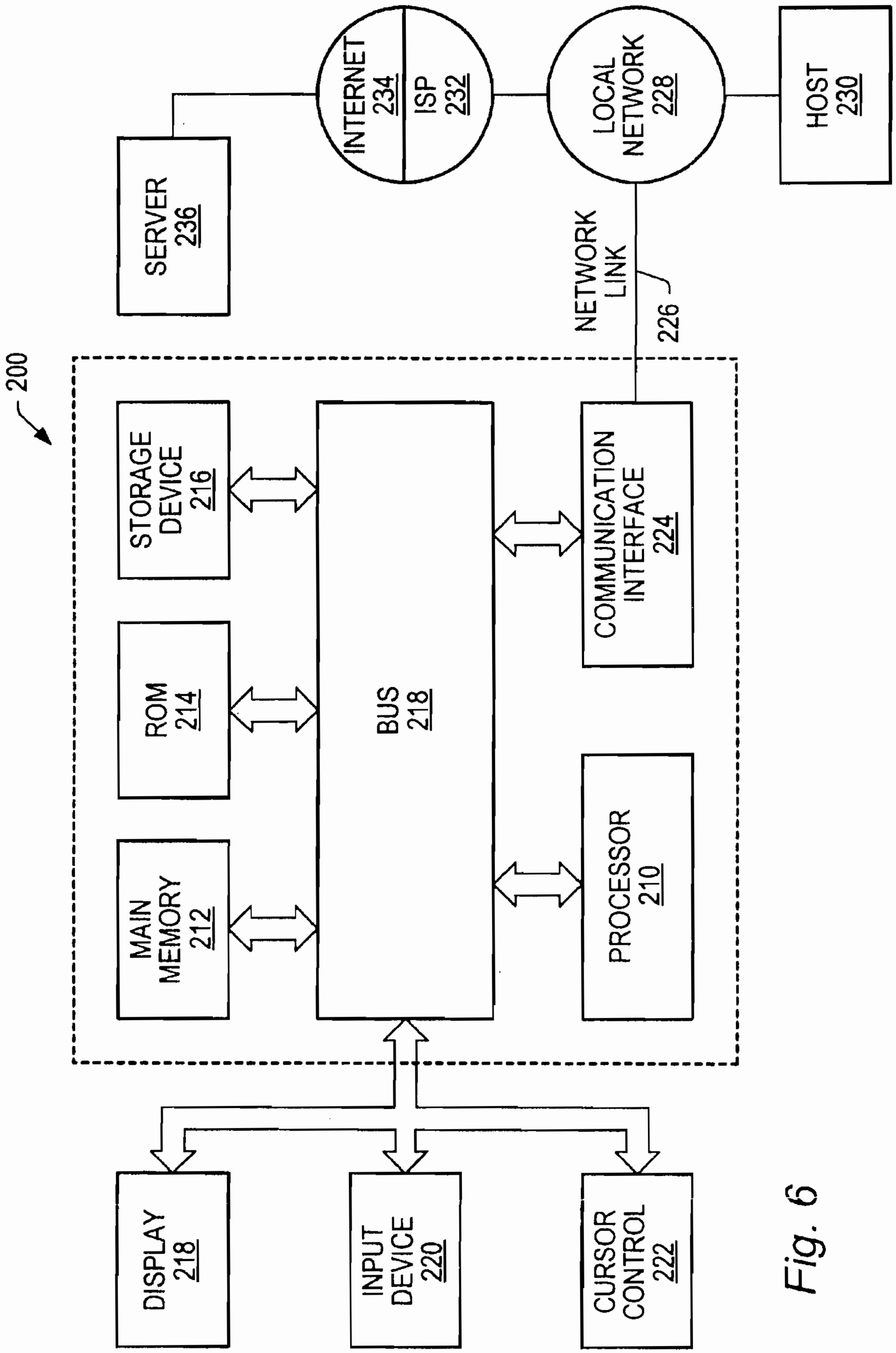

200
MAIN MEMORY 212
ROM 214
STORAGE DEVICE 216
SERVER 236
INTERNET 234
ISP 232
BUS 218
LOCAL NETWORK 228
HOST 230
NETWORK LINK
226
COMMUNICATION INTERFACE 224
PROCESSOR 210
DISPLAY 218
INPUT DEVICE 220
CURSOR CONTROL 222

Fig. 6

DISTINGUISHED NAME SCOPING SYSTEM FOR EVENT FILTERING

CROSS REFERENCE TO RELATED PATENT APPLICATIONS

This patent application is related to other patent applications, filed herewith on the same day and entitled "Apparatus, Methods And Computer Program Products For Network Management Operations Relating To Network Management Protocol Adapter Security Software (MPASS) For Single And Multiple Users", Ser. No. 09/330,902, "Secure User Association And Set-Up Using Network Management Protocol Adapter Security Software (MPASS)", Ser. No. 09/330,932, "Messaging With User Name Access Identification Using Network Management Protocol Adapter Security Software (MPASS)", Ser. No. 09/330,521, "Independent Log Containment Hierarchy", Ser. No. 09/330,514, and "Domain Access Control For Logging Systems", Ser. No. 09/332,270. These related patent applications are hereby expressly referenced and incorporated herein in their entirety.

COPYRIGHTS IN PATENT MATERIAL

TECHNICAL FIELD

The field of this application relates to apparatus, methods and computer program products relating to network management operations and protocol adapter security software.

BACKGROUND OF THE INVENTION

Network management is performed by network carriers and operators to ensure that mission-critical networks are continually operating normally, without service-affecting problems. Accordingly, network management platforms, such as the Sun Microsystems Solstice Enterprise Manager (SEM) are employed. The SEM is a management framework that complies with Telecommunications Management Network (TMN) standards, as defined by the International Telecommunications Union (ITU). To ensure that an operator's network is functioning properly, it is necessary to monitor or listen for information on events that might indicate network status changes. For example, failure of a network switching device or break in a circuit may produce an event that is received and processed by the SEM network management platform.

An "event" may be defined as a signal, or the underlying occurrence, indicating that one or more changes has occurred in the state of an entity or device on the network. Event signals may include a communications alarm signal (indicating that a device has come on-line, has gone off-line or has developed a problem), an equipment alarm signal (occurrence of an error state), a quality-of-service alarm signal (deterioration of the strength or resolution or throughput of a signal or group of signals), security alarm signal (indicating that unauthorized access has been detected), and an attribute change (indicating that data for a device or process are no longer available, for reasons other than occurrence of an error).

To control and coordinate the software associated with a network of computers and peripherals, network managers often employ special purpose software designed to track, establish communications with and control other software entities and processes that represent one or more network reporting devices (referred to as "agents") or that exist independently. Software used by the network manager interacts with various platform-level software services to allow the network manager to locate and interact with other entities running on the network. Entities on the network can communicate with each other and with a network manager by sending and receiving messages with agreed-upon formats. A message can be a request, a response or an event signal. An entity running on the network may "subscribe to" notifications of events generated by other entities so that a cooperative relationship between the entities can be maintained. Given the number and variety of events that can occur, the volume of event notifications processed by the network management software can be very large, even when few or no error messages are being transmitted. This volume can reduce system performance dramatically.

Typically, a computer network will rely upon a single, centralized service to manage, process and/or monitor the network communications. This reliance upon a single service to process such high volumes of data creates a risk of catastrophe or collapse if the central service fails, even with swift recovery. These risks are not acceptable for large-scale networks that must be available at all hours, seven days per week. Further, if all event notifications must be processed and analyzed by each network operator or monitor, each such operator may have to provide enormous computing power for this purpose, even where the amount of information of interest to an operator is small.

During an associated event processing activity, appropriate operators are notified and corrective actions may be taken. In a large network, events occur very frequently, perhaps on the order of hundreds of events per second. Accordingly, efficient event notification processing and distribution is a key to successful network management platform operation. Currently, events are characterized by event type, indicating the nature of the event, to the extent that this information is determinable. Possible event types are defined in the managed object's management information base (MIB). By subscribing to particular types of events, a network manager can receive notifications of events of a particular type. When a network manager subscribes to several notifications of several types of events, an event filtering mechanism or discriminator is implemented, using common management information system (CMIS) filtering to ensure that only notifications of selected event types are forwarded for consideration by the subscribing network manager.

However, there is currently no mechanism permitting a network manager to receive notifications concerning events associated with a specified source. It is not possible for a network manager to receive information focusing on a selected managed object or objects. However, it is frequently desirable for a network manager to be able to subscribe only to events from particular sources or objects. For example, a network management operator responsible for cellular switches in and around Frankfurt might only want to receive event notifications concerning the Frankfurt portion of the network. This is a practical, long-felt need of network operators throughout the world, as well as in the United States. Further, it is desirable that the event notification processing be scalable to correspond to the volume that is anticipated under given circumstances on a network.

SUMMARY OF THE INVENTION

This invention makes it possible to subscribe to notification of events that arise or occur at specified sources or

specified objects. This makes it easier for a Solstice Enterprise Manager (SEM) to focus on the portion of a network of interest to that operator and reduces the burden on an SEM application developer, who would otherwise have to subscribe to all event notifications from all sources, and then to use custom code to laboriously filter or screen out events from sources or objects that are of no interest to the developer. According to the invention, the SEM operator may specify one or more event characteristics and/or one or more levels of objects associated with a computer network. An object may have associated with it one or more attributes or characteristics, such as an event type, the location (node or group of nodes) or region where an event occurred, a date and/or time interval during which one or more events of interest occurred, a type of component or device that was affected by an event, and other similar attributes. With this specification in place, only events that arise from one or more specified levels of objects and have specified characteristics are registered for consideration by that developer.

The SEM infrastructure is enhanced, according to the invention, to permit filtering internally, according to the source or object associated with an event. A network management operator thereafter receives only event notifications only from one or more specified sources and, optionally, only as to specified types of events. Accordingly, a sophisticated and improved capability is provided to allow network operators to more precisely specify notifications of events to be received. This improves the ability to focus on the portion(s) of the network system of particular concern to the operator.

The SEM has a distributed client-server architecture in which clients or applications use the services offered by the server or platform. One service offered by the platform is subscription for event notifications from network agents managed by the platform, based on one or more filtering criteria. An event notification distribution subsystem (EDS), according to the invention, allows transmission of an event notification to an event subscriber, an application that monitors network communications for event notifications (ENs) in which the event subscriber is interested. The event subscriber specifies a discriminator mechanism or CMIS filter, which is written in a predetermined CMIS filter specification. The event source includes an application or service entity that issues an EN that will be received by one or more identified event subscribers, if certain characteristics or attributes of an underlying event agree with attributes specified by an event discriminator.

However, this type of CMIS filter is of limited flexibility and does not permit specification of, and subscription for, notification of "wild card" events associated with one or a group of distinguished names (DNs). The present invention permits an application to receive notifications of all events whose objects belong to one or more specified levels of objects. A managed object may agree with a given data network prefix, which may include a specification of site, channel and/or element. For example, if the prefix is set to SiteId=5, ENs corresponding to all events with SiteId=5 and ChannelId=DC (don't care) and ElementId=DC will be received and registered by the EN application.

According to the invention, an EN received by an application passes its DN prefix for scoping, in addition to specific information concerning characteristics of the underlying event. Use of a DN scoping mechanism allows reduction in the number of fan-out events within the EDS, because an application only registers for events that are likely to be of common interest. Thus, the present invention permits receipt of ENs for all events whose managed object instances (MOIs) agree with a specified DN prefix. DN filtering may be implemented by specifying a DN scope and at least one DN attribute.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates an environment in which the invention can be used.

FIG. **2** illustrates some of the object levels that can be specified according to the invention.

FIG. **3** illustrates three well known layers of a network.

FIG. **4** illustrates event processing according to the invention.

FIG. **5** illustrates a suitable set of interrogatories that can be used to specify the event filtering to be used according to the invention.

FIG. **6** is a schematic view of a computer system.

DESCRIPTION OF PREFERRED MODE OF THE INVENTION

FIG. **1** illustrates general operation of an event notification distribution system (EDS) according to the invention. A user **11** communicates with a discriminator mechanism (DM) **13**, which may include a network agent that communicates with and monitors the communications that pass across a network **15**. The network **15** includes a plurality of nodes or other network participants, **17-1**, **17-2**, **17-3**, that communicate with each other using the network. One or more nodes **17**-i (i=1, 2, 3, . . .) reports from time to time on certain events that have occurred at that node or that affect that node, usually by transmission of an event notification that contains relevant information describing or characterizing the event to one or more other nodes and/or to a central reporting station.

For example, a crosspoint switch in an asynchronous transport mode (ATM) network may transmit a notification of congestion upstream from that switch, using an upstream control signal, and/or downstream from that switch, using a downstream control signal, to request that other switches reduce the rate at which communications are transmitted to the congested switch. Alternatively, a new node may join or rejoin the network **15** and may transmit certain interrogation signals informing other nodes of certain characteristics of the new node and attempting to determine the location of the new node.

The DM **13** is preferably located on or adjacent to the network **15** so that any communication concerning an event that occurs at or affects a specified node or group of nodes on the network will pass by or through the DM. The DM **13** listens promiscuously to, and analyzes, all communications, such as ENs, that pass by or through itself. When the DM **13** determines that a given communication concerns one or more objects in a specified level, or group of levels, of objects for which the DM has responsibility, the DM temporarily or permanently records the communication in a memory unit **19**. Optionally, a transmitter **21** associated with the DM **13** and/or the memory unit **19** subsequently transmits this recorded information to all nodes or other devices on or adjacent to the network **15** that have subscribed to events that arise at or concern the specified objects.

Alternatively, the DM **13** may transmit information on the event communication directly to the network management operator (NMO) without delay. The DM **13** thus filters the communications in terms of specified objects and object levels and (optionally) and in terms of one or more specified event characteristics, such as event type, event location, time

and/or date of event occurrence, and/or type of component or device involved in the event. DM filtering may occur by scanning each passing communication for any mention, in a data header, the data itself or a data trailer, of an object or class of objects or event characteristics for which the DM 13 is directed to scan. This DM filtering and reporting will occur independently of whether the reported communication is successfully received by its intended recipients on the network.

FIG. 2 illustrates a management information tree (MIT) having several levels of objects. At the top of the tree is a root level, from which all other levels descend. Immediately below the root level, and connected thereto, is a systemId level that can serve as a level zero for related objects. The next lower level ("level one") in the example shown in the diagram has three objects, with associated nodes indicated respectively as **node1**, **node2** and **node3**. The next lower level ("level two") has five objects, indicated as **node1\1** and **node1\2** (both connected to **node1** in level one), **node 2\1** (connected to **node2**), and **node 3\1** and **node3\2** (connected to **node3**). The next lower level ("three") has 11 objects, indicated as nodes **1\1\1**, **1\1\2** and **1\1\3** (connected to node **1\1** in level two), nodes **1\2\1** and **1\2\2** (connected to node **1\2**), nodes **2\1\1** and **2\1\2** (connected to node **2\1**), nodes **3\1\1**, **3\1\2** and **3\1\3** (connected to node **3\1**), and nodes **3\2\1** and **3\2\2** (connected to node **3\2**). Additional levels of objects can be added and identified in a similar manner.

Beginning from the root node at the top of the tree in FIG. **2**, one can specify a path to reach a particular object or node. For example, a path to reach the object associated with the node **3\1\2** can be specified as

{nodeId="root"/nodeId="systemId"/nodeId="node3"/ nodeId="node3\1"/nodeId="3\1\2"}

The invention and the object/node specification notation discussed here allows an operator to specify ENs associated with: (1) one or more objects, such as **node31/2** and node **2\1\1**; (2) a given level of objects, such as node **1\1**, node **1\2**, node **2\1**, node **3\1** and node **3\2**; (3) all objects between and including a first specified level and a second specified level (e.g., between levels two and five); (4) all objects at lower levels connected to a given object, such as all lower level objects connected to node **1\2**; and other suitable specifications of objects based on object levels.

Computer network software and hardware operations can be organized conceptually into an application or user layer, a platform layer and a hardware layer. As illustrated in FIG. **3**, the application layer **101** in a network **100** includes user interface (UI) software **103**, which allows a network user to communicate with software and hardware installed on the network. A user can send requests, receive responses, reply to requests, receive alarm signals, receive status reports, locate and control devices installed on the network. The hardware layer **121** includes the physical devices **123** installed on the network. Examples of these devices include computers, communications devices, bridges, routers, gateways, servers, hubs, modems, printers, display screens, scanners and network interface cards.

The platform layer **105**, located between and communicating with, the application layer **101** and the hardware layer **121**, includes network management software that allows a network administrator, operating an application such as the UI **103**, to obtain access to, and to provide communication between, entities and devices on the network. Network communications, including event notifications, may be handled by a centralized management information server (MIS) **107** that coordinates messages sent between entities and devices operating on each of the network layers. The MIS **107** interacts with a DN server **109**, connected to an MIT **110**, that provides a database including names and network addresses for all entities and devices installed on the network. A topology service **111** and a logging/alarm service **113** provide resources for managing network entities, devices and alarms and for keeping track of any changes in network topology.

The platform layer may further include one or more agents **115** and a message protocol adapter (MPA) **117** that allow communication between one or more devices **123** in the hardware layer **121** and the MIS **107**. An event notification distribution system (EDS) **119**, as described above and preferably located in the platform layer **105**, communicates with the MIS **107**, the topology server **111**, the logging/alarm server **113** and the UI **103**. The EDS **119** processes event notifications and transmits these signals to various client or listener entities on the network. In a first embodiment, the EDS **119** runs on a separate processor and reduces computational overhead associated with the MIS **107**. Using one processor for the MIS **107** and a second processor for the EDS **119** reduces the likelihood that the MIS and the EDS will fail or malfunction simultaneously. In a second embodiment, the EDS **119** operates as a separate process through the MIS **107**. In a third embodiment, the EDS **119** operates as part of the same process within the MIS **107**.

Initiation of event notification is illustrated in a flow chart in FIG. **4**. At step **131**, an event notification is generated on the network. At step **133**, the EN is processed, using any suitable intermediary, such as an MPA, that operates between the event notification generator **131** and an EDS and/or an MIS. The intermediary directs the EN to an EDS and/or to an MIS, using several criteria, such as object level(s) associated with each mentioned DN, source and time/date of the event, severity of the event, type of intermediary and type of client-listener. The system then determines, in step **135**, if the EN information should be sent initially to the EDS.

If the answer to the question in step **135** is "yes", the EDS receives and processes the EDS-relevant information for event reporting, in step **137**, and sends any MIS-relevant information to the MIS, in step **139**, for further processing. The system then moves to step **145**.

If the answer to the question in step **135** is "no", the MIS receives and processes the MIS-relevant information for event reporting, in step **141**, and sends any EDS-relevant information to the EDS, in step **143**, for further processing. The system then moves to step **145**.

In step **145**, the system sends (or temporarily holds for later transmission) EN information on relevant events, if any, to the appropriate event reporting subscribers on the network and recycles to step **133** (or step **131**) to receive another incoming EN.

Alternatively, the EDS can process all ENs. This approach will often promote efficiency in EN signal handling but will not provide any redundancy if the EDS fails or malfunctions.

Presently, a network operator may subscribe for, and receive notifications of, all network events classified by event type, using a discriminator mechanism. Specified event type or types might include congestion, erratic component operation, component failure, error rate outside prescribed bounds, and other similar types of events that directly or indirectly reflect or affect network operations. The present invention allows a network operator to subscribe for, and receive notifications of, all events occurring at or affecting specified levels of objects and having specified attributes. Other attributes associated with an object

may include a specified geographic location or region where the event occurred, a specified class of components or devices (e.g., switches or buffers), a class of events that occur or are initiated in a specified time interval or upon a specified date, and event severity, among other things.

Filtering based on DN scoping requires specification of DN scope (e.g., which object or object level or levels are involved). This DN filtering optionally works with and operates on top of existing (CMIS) filtering that is already in place, such as EN filtering based on event type.

For example, a DM **13** may be directed to scan for SiteId, ChannelId, ElementId, date and/or time of occurrence of event, type of device, component or node ("source"), and location of source within a prescribed region, as well as for type of event, according to the invention.

FIG. **5** illustrates a suitable sequence of interrogatories that can be presented to the NMO in order to specify event filtering according to the invention. These interrogatories allow an NMO to specify an object level or group of object levels for which event notifications are to be reported. These interrogatories also allow an NMO to specify a node or group of nodes (event sources) for which events are to be specified and/or to specify a geographical region (individual nodes unspecified). The NMO may also specify a date and/or time interval for which event occurrences are to be reported. The NMO may also specify a type or types of devices involved in an event and/or an event type for which event reporting is required. If a particular event characteristic is not of concern to the NMO, the "Characteristic" column would be left blank and the "Don't Care" column would be marked with an "X" or some other suitable symbol. Parameters that are explicitly specified are treated as subset specifications in a Boolean intersection. Specification of objects and characteristics may also be done by any other suitable means of data entry.

For example, an event filter interrogatory that specifies (S1) object levels **2–4**, (S2) event sources within a geographical region R, and (S3) events occurring within a time interval $t1 \leqq t \leqq t2$, will filter and report only those events that lie in all of the specified subsets (S1), (S2) and (S3); all other parameters are ignored in this event filtering. The set of interrogatories shown in FIG. **5** is not intended to limit the event characteristics that can be specified according to the invention. One or more of the event characteristics shown in FIG. **5** can be optionally deleted.

For example, an NMO responsible for operation of all cellular telecommunications switches in the Frankfurt region might only want to receive ENs for cellular network switches (type of source) in the Frankfurt region (source location). At a particular time of day or time of the week, this NMO might wish to focus on a particular class of problems, such as signal congestion during the hours of 10:00 to 4:00, Frankfurt time, as communicating networks in other time zones also become more active. In this example, the NMO can subscribe for ENs based on event source location, event source type, day and/or time interval of event occurrence, and event type. The event filtering and EN is performed elsewhere for the NMO, based on operator specifications, so that the operator need not perform the filtering operations itself.

An NMO may be more concerned with a first class of object levels and event characteristics during a first time interval and be more concerned with a second class of object levels and event characteristics during a second time interval. For example, signal traffic congestion may be more important during peak traffic hours, and the results of off-peak, network-initiated source testing may be of greater interest during certain off-peak hours. The invention allows for this by permitting the NMO to modify its list of specified object levels and event characteristics from one time interval to another.

Using the invention described here, it is possible for an NMO to specify and receive reports on events specified by object level or levels and characterized by one or more event attributes, such as event type, event source, event severity, event source location, event component or device type, date and/or time that event occurred, and for any combination of these object levels and event characteristics. The invention provides an improved capability for an NMO to specify, closely monitor and analyze only the class of object levels and event characteristics that are of direct concern to the NMO.

FIG. **6** shows a block diagram of a general computer system **200**, which may be used to implement various hardware components of the invention, such as a client, an applications server and a database management system. The computer system **200** includes a bus **208** or other communication mechanism for communicating information and a processor **210**, coupled with the bus **208**, for processing information. The computer system **200** also includes a main memory **212**, such as a random access memory (RAM) or other dynamic storage device, coupled to the bus **208**, for storing information and instructions to be executed by the processor **210**. The main memory **212** also may be used for storing temporary variables or other intermediate information during execution of instructions by the processor **210**. The computer system **200** further optionally includes read only memory (ROM) **214** or other static storage device, coupled to the bus **208**, for storing static information and instructions for the processor **210**. A storage device **216**, such as a magnetic disk or optical disk, is provided and is coupled to the bus **208** for storing information and instructions.

The computer system **200** may also be coupled through the bus to a display **218**, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device **220**, including alphanumeric and other keys, is coupled to the bus for communicating information and commands to the processor **210**. Another type of user input device is a cursor control **222**, such as a mouse, a trackball or cursor direction keys for communicating direction information and command selections to the processor **210** and for controlling cursor movement on the display **218**. This input device typically has one degree of freedom in each of two axes, such as x- and y-axes, that allows the device to specify locations in a plane.

The functionality of the invention is provided by the computer system **100** in response to the processor **210** executing one or more sequences of instructions contained in main memory **212**. These instructions may be read into main memory **212** from another computer-readable medium, such as a storage device **216**. Execution of the sequences of instructions contained in the main memory **212** causes the processor **210** to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of, or in combination with, software instructions to implement the invention. Embodiments of the invention are not limited to any specific combination of hard-wired circuitry and software.

The term "computer-readable medium", as used herein, refers to any medium that participates in providing instructions to the processor **210** for execution. This medium may take many forms, including but not limited to non-volatile media, volatile media and transmission media. Non-volatile

media includes, for example, optical and magnetic disks, such as the storage disks **216**. Volatile media includes dynamic memory **212**. Transmission media includes coaxial cables, copper wire and fiber optics and includes the wires that are part of the bus **208**. Transmission media can also take the form of acoustic or electromagnetic waves, such as those generated during radiowave, infrared and optical data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, a hard disk, magnetic tape or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes or apertures, a RAM, a ROM, a PROM, an EPROM, a Flash-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can be read.

Various forms of computer-readable media may be involved in carrying out one or more sequences of one or more instructions to the processor **210** for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone, using a modem. A modem local to the computer system **200** can receive data over a telephone line and use infrared transmitter to convert and transmit the data to the an infrared detector connected to the computer system bus. The bus will carry the data to the main memory **212**, from which the processor receives and executes the instructions. Optionally, the instructions receive by the main memory **212** can be stored on the storage device **216**, either before or after execution by the processor **210**.

The computer system **200** also includes a communications interface **224**, coupled to the bus **208**, which provides two-way data communication coupling to a network link **226** that is connected to a local area network (LAN) or to a wide area network (WAN). For example, the communications interface **224** may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, the communications interface **224** may be a local area network card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, the communications interface **224** sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

The network link **226** typically provides data communication through one or more networks to other data devices. For example, the data link **226** may provide a connection through an LAN **228** to a host computer **230** or to data equipment operated by an Internet Service Provider (ISP) **232**. The ISP, in turn, provides data communication services through the world wide packet data communication network, now commonly known as the "Internet" **234**, served by one or more servers **236**. The LAN **228** and the Internet **234** both use electrical, electromagnetic and/or optical signals to carry the digital data streams. The signals carried by these network, the signals carried on the network link **226** and the signals carried on the communications interface **224**, are examples of carrier waves that transport the information.

What is claimed is:

1. A computer implemented method of managing operations on a network, the method comprising:

receiving on a network an event communication that characterizes an event that has occurred at, or that affects, one or more network participants;

examining the event communication to determine whether the event is associated with at least one object or object level on a selected object list having at least one specified object or object level; and when the event is associated with at least one object or object level that is on the object list, taking at least one of two actions: (1) communicating selected information that describes the event to a selected event information recipient; and (2) storing selected information on the event; and

examining said event communication to determine whether said event has at least one event characteristic that appears on a selected event list of at least one specified event characteristic; and when said event has at least one characteristic that is on the characteristic list, taking at least one of two actions: (1) communicating said selected information on at least one selected event characteristic that describes said event to a selected event information recipient; and (2) storing said selected information on said event; wherein said characteristic list is chosen to include at least one of the following event characteristics: one or more specified network nodes involved in said event; a specified geographical region in which said event occurs; a specified time interval within which said event occurs or is initiated; a specified class of devices involved in said event; specified severity of said event; and said event is one or more specified event types.

2. The method of claim **1**, further comprising: when said event does not have at least one characteristic that is on said characteristic list, taking no action on said event communication.

3. The method of claim 1, further comprising storing said selected information on said event having at least one characteristic that is on said characteristic list, and communicating said selected information on said at least one selected event characteristic to said selected event information recipient at a selected later time.

4. The method of claim **1**, further comprising modifying said characteristic list at least once for a selected reporting time interval.

5. The method of claim **1**, further comprising modifying said object list at least once for a selected reporting time interval.

6. A computer implemented system for managing operations on a network, the method comprising a computer that is programmed:

to receive on a network an event communication that characterizes an event that has occurred at, or that affects, one or more network participants;

to examine the event communication to determine whether the event is associated with at least one object or object level on a selected object list having at least one specified object or object level; and when the event is associated with at least one object or object level that is on the object list, to take at least one of two actions: (1) communicate selected information that describes the event to a selected event information recipient; and (2) store selected information on the event; and

to examine said event communication to determine whether said event has at least one event characteristic that appears on a selected event list of at least one specified event characteristic; and when said event has at least one characteristic that is on the characteristic list, to take at least one of two actions: (1) communicate said selected information on at least one selected event characteristic that describes said event to a selected event information recipient; and (2) store said selected

information on said event; wherein said selected characteristic list is chosen to include at least one of the following event characteristics: one or more specified network nodes involved in said event; a specified geographical region in which said event occurs; a specified time interval within which said event occurs or is initiated; a specified class of devices involved in said event; specified severity of said event; and said event is one or more specified event types.

7. The system of claim 6, wherein, when said event does not have at least one characteristic that is on said characteristic list, said computer is programmed to take no action on said event communication.

8. The system of claim 6, wherein said computer is further programmed: to store said selected information on said event having at least one characteristic that is on said characteristic list; and to communicate said selected information on said at least one selected event characteristic to said selected event information recipient at a selected later time.

9. The system of claim 6, wherein said computer is further programmed to modify said characteristic list at least once for a selected reporting time interval.

10. The system of claim 6, wherein said computer is further programmed to modify said object list at least once for a selected reporting time interval.

11. A computer program product embodied in a computer usable medium having a computer readable code mechanism embodied therein for managing transactions, the computer program product comprising:

a first computer readable code mechanism configured to receive on a network an event communication that characterizes an event that has occurred at, or that affects, one or more network participants;

a second computer readable code mechanism configured to examine the event communication to determine whether the event is associated with at least one object or object level on a selected object list having at least one specified object or object level;

a third computer readable code mechanism configured so that, when the event is associated with at least one object or object level that is on the object list, to take at least one of two actions: (1) communicate selected information on the event that describes the event to a selected event information recipient; and (2) store selected information on the event; and

wherein at least one of said first, second and third code mechanisms is configured: to examine said event communication to determine whether said event has at least one event characteristic that is on said characteristic list; and when said event has at least one characteristic that is on said characteristic list, to take at least one of two actions: (1) communicate said selected information on at least one selected event characteristic that describes said event to a selected event information recipient; and (2) store said selected information on said event; wherein said selected list associated with said second code mechanism includes at least one of the following event characteristics: one or more specified network nodes involved in said event; a specified geographical region in which said event occurs; a specified time interval within which said event occurs or is initiated; a specified class of devices involved in said event; specified severity of said event; and said event is one or more specified event types.

12. The computer program product of claim 11, wherein, when said event does not have at least one characteristic that is on said characteristic list, said third code mechanism takes no action on said event communication.

13. The computer program product of claim 11, wherein at least one of said first, second and third code mechanisms is further configured: to store said selected information on said event having at least one characteristic that is on said characteristic list: and to communicate said selected information on said at least one selected event characteristic to said selected event information recipient at a selected later time.

14. The computer program product of claim 11, at least one of said first, second and third code mechanisms is further configured to modify said characteristics list of said at least once for a selected reporting time interval.

15. The computer program product of claim 11, at least one of said first, second and third code mechanisms is further configured to modify said object list of said at least once for a selected reporting time interval.

* * * * *